Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matic Kunaver

Samodejno prepoznavanje obraznih izrazov na iOS platformi

na univerzitetnem študijskem programu
računalništva in informatike

Mentor: doc. dr. Peter Peer

Ljubljana 2014

Rezultati diplomskega dela so intelektualna lastnina avtorja. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Najprej preučite pristope za samodejno prepoznavanje človeških obraznih izrazov na slikah. Implemenirajte pristop, ki se bo približal najboljšim rezultatom na tem področju. Evalvacijo naredite nad ustrezno bazo. Na koncu pripravite knjižnico, ki bo omogočala uporabo pristopa na napravah z iOS operacijskim sistemom.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matic Kunaver, z vpisno številko **63080008**, sem avtor diplomskega dela z naslovom:

Samodejno prepoznavanje obraznih izrazov na iOS platformi

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Petra Peera,
- so elektronska oblika diplomskega dela, naslov (slov.), povzetek (angl.) ter ključne besede (slov.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 23. junij 2014 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Petru Peeru za strokovno pomoč pri izdelavi diplomske naloge.

Zahvala gre tudi moji družini, ki me je spodbujala in pomagala pri premoščanju vseh težav v času študija.

Kazalo

Povzetek

V tem delu je predstavljen način za samodejno prepoznavanje človeških obraznih izrazov s slik, pridobljenih s prenosnim telefonom, ki ima naložen operacijski sistem iOS. Prepoznavanje obraznih izrazov je proces, sestavljen iz več korakov. V prvem koraku je treba določiti območje obraza. Nato je treba preoblikovati podatke v računalniku primernejšo obliko. Za to obstaja več različnih metod, v tem delu pa je uporabljena in tudi bolj podrobno predstavljena metoda lokalnih binarnih vzorcev. V zadnjem koraku je treba preoblikovane podatke shraniti in izgraditi model za napovedovanje obraznih izrazov. Za napovedovanje je uporabljena metoda podpornih vektorjev, ki ima dobro razmerje med hitrostjo in natančnostjo. Končni rezultat tega dela je knjižnica, ki jo lahko vsak razvijalec priloži svojemu projektu in s pripravljenimi programskimi funkcijami dobi možnost za prepoznavo obraznih izrazov. Razvita rešitev je bila naučena in preizkušena na označenih obrazih iz baze Cohn-Kanade in iz baze tekmovanja s spletne strani Kaggle. Vsi algoritmi v tem delu se opirajo na programsko knjižnico OpenCV, razvita knjižnica pa deluje na večini prenosnih naprav družbe Apple z operacijskim sistemom iOS. Knjižnica je primerna za prepoznavanje obraznih izrazov na eni sami sliki ali na video posnetkih, prepozna pa lahko do osem različnih izrazov (če štejemo tudi nevtralni obrazni izraz).

Ključne besede: računalniški vid, strojno učenje, obrazni izrazi, prenosni telefoni, preoblikovanje podatkov računalniški vid, strojno učenje, obrazni izrazi, prenosni telefoni, preoblikovanje podatkov.

Abstract

In this diploma thesis we present an algorithm for automatic recognition of facial expressions in images from mobile devices with iOS operating system. Facial expression recognition is a process, consisting of several steps. In the first step we have to find a face region. In the next step we need to transform input data into a computer readable form. There are several methods for transforming the data. We used a method called Local Binary Pattern transformation. In the last step the data has to be stored and a computer model for predicting facial expressions has to be generated. Prediction is done by using Support Vector Machines. SVM method is fast and reliable. The final result of the work presented in the thesis is a library available to all developers. The library is easy to use and enables developers to get facial expression information with almost zero work. This solution was tested on labeled faces from Cohn-Kanade database and on labeled faces from Kaggle website. All algorithms in this work rely on OpenCV framework. The library works on almost all Apple mobile devices with iOS operating system. Library can detect facial expressions in a single photograph or in a video. It can detect up to eight different facial expressions (together with neutral expression).

Keywords: computer vision, machine learning, facial expressions, mobile phones, data transformation computer vision, machine learning, facial expressions, mobile phones, data transformation.

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	application programming interface	programski vmesnik
AU	action units	akcijske enote
FACS	facial expression coding system	sistem za kodiranje obraznih izrazov
k-NN	k-Nearest Neighbors algorithm	algoritem k najbližjih sosedov
LBP	local binary pattern	lokalni binarni vzorec
RBF	radial basis function	radialna bazna funkcija
SVM	support vector machine	metoda podpornih vektorjev

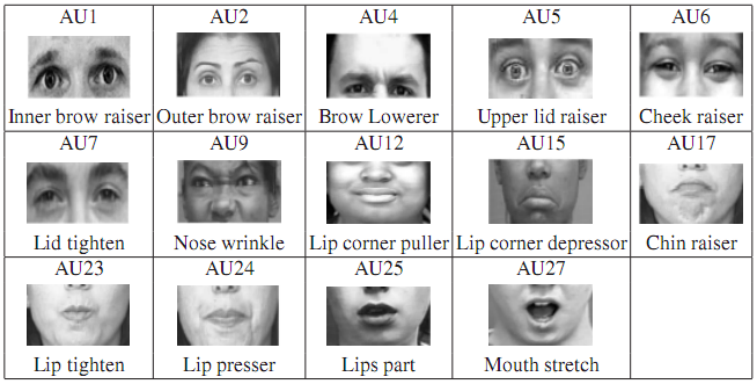
Poglavje 1

Uvod

Obrazni izrazi so spremembe obraza, ki se odrazijo pri spremembah človeških notranjih čustvenih stanj, pri različnih namenih ali pri socialnih interakcijah. Analiza človeških čustev je danes predmet različnih raziskovanj. Velikokrat ljudje obrazne izraze zamenjujejo za človeška čustva. Ta dva pojma ne pomenita istega. Za čustva potrebujemo višji nivo znanja. Čustva izražajo tudi druge kognitivne procese, fizične napore in medsebojne človeške odnose. Pri interpretaciji obraznih izrazov pa nas zanima samo izraz na obrazu, brez podatkov o spolu, kulturi ali kontekstu.

Samodejna prepoznava obraznih izrazov je zaradi tega zanimiva in težka tematika, ki je dandanes vedno bolj zaželena in razširjena. Čeprav je danes razvitih že zelo veliko različnih metod in načinov za samodejno prepoznavo obraznih izrazov, pa so rešitve z visoko natančnostjo še vedno precej redke zaradi kompleksnosti in velike variance človeških čustev.

Za opis obraznih izrazov obstaja več različnih pristopov. Najbolj razširjen znanstveni pristop, kodni sistem obraznih izrazov (FACS – Facial Expression Coding System), je bil razvit z namenom enostavnejše in hitrejše klasifikacije izrazov, kot je bila mogoča v prejšnjih sistemih. Temelji na frekvenci, intenzivnosti, valentnosti in trajanju obraznega izraza. Obrazni izraz je določen kot sprememba obraza iz nevtralnega obraznega izraza (brezizrazni obraz) v ne-nevtralni izraz in nato ponovno nazaj. Za opis se uporabljajo akcijske enote (angl. AU – Action units), ki so osnovne akcije posameznih mišic ali skupin mišic. Nekaj akcijskih enot je prikazanih na sliki 1.1. S temi enotami se lahko določi vseh sedem osnov-



Slika 1.1: Prikaz akcijskih enot

nih obraznih izrazov, ki prikazujejo veselje, žalost, presenečenje, strah, jezo, gnus in prezir [3, 4].

Prvi in najpomembnejši korak pri prepoznavanju obraznega izraza je preoblikovanje podatkov v računalniku prijaznejšo obliko. Ko imamo podatke pravilno preoblikovane, je treba te podatke shraniti v računalniku primerno obliko in sestaviti model, ki bo primeren za napovedovanje trenutnega obraznega izraza. Najbolj enostaven način je, da primere enostavno shranimo in pri napovedovanju poiščemo primer z najmanjšim odstopanjem. Ker pa je to precej prostorsko in časovno zahtevno, lahko uporabimo druge algoritme strojnega učenja. Najpogosteje se uporabljajo SVM-klasifikatorji. Uporabni modeli pa so tudi metoda najbližjih sosedov, umetne nevronske mreže, naivni Bayes in ostale.

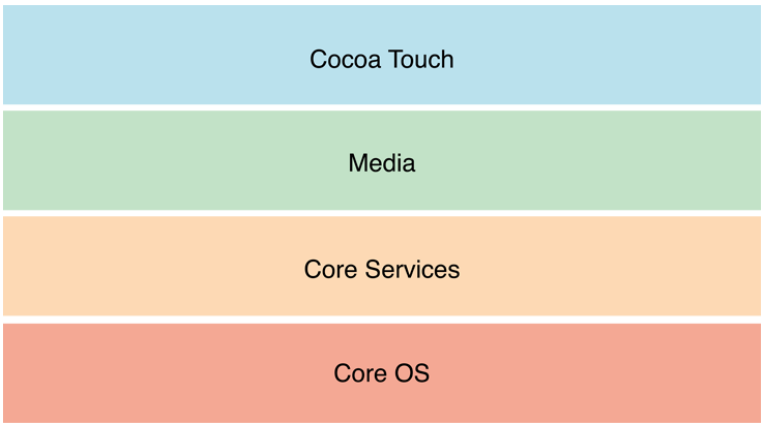
1.1 Predstavitev uporabljenih tehnologij

Vsi algoritmi in programske rešitve, uporabljene v tem delu, temeljijo na obstoječih tehnologijah, brez katerih tako hiter razvoj ne bi bil mogoč. Grafični vmesnik je prirejen za operacijski sistem iOS 7 in uporablja osnovna programska ogrodja (angl. framework) družbe Apple. Vsi algoritmi strojnega učenja in detekcije obrazov pa temeljijo na odprtokodni knjižnici OpenCV, ki je napisana v programskem jeziku C in C++ [1, 2].

1.1.1 Predstavitev operacijskega sistema iOS

Na prenosnih napravah družbe Apple je naložen operacijski sistem iOS. iOS temelji na namizni različici operacijskega sistema Mac OS X. Z njim si deli jedro Darwin in številne knjižnice. V iOS 7, najnovejši različici operacijskega sistema, so štirje abstrakcijski nivoji (slika 1.2): nivo Core OS, nivo Core Services, nivo Media in nivo Cocoa Touch [5].

- Nivo Core OS vsebuje nizkonivojske lastnosti, na katerih so zgrajene številne tehnologije, ki jih uporabljajo drugi nivoji.
- Nivo Core Services vsebuje ključne sistemske storitve za aplikacije. Tu se nahajajo tehnologije, ki podpirajo iCloud, socialne storitve in omrežja.
- Nivo Media vsebuje grafične, zvočne in video tehnologije, ki se uporabljajo za večpredstavnostne zmožnosti v aplikacijah.
- Nivo Cocoa Touch vsebuje ključne knjižnice za razvoj iOS aplikacij.



Slika 1.2: Nivoji operacijskega sistema iOS

Za iOS platformo družba Apple podpira in priporoča uporabo programskega jezika Objektni-C (v skupni kodi se lahko piše tudi v jeziku C in C++). Uradno razvojno okolje se imenuje Xcode in omogoča razvoj, razhroščevanje ter distribucijo aplikacij iOS in Mac.

Za razvoj je nujno potrebna tudi fizična naprava s kamero (iPhone, iPad ali iPod touch), saj vgrajeni simulator ne podpira zajemanja iz kamere. Testiranje na sami napravi pa je možno samo ob nakupu enoletne naročnine (v času pisanje 99 $), ki jo ponuja družba Apple vsem razvijalcem. Ob nakupu naročnine dobijo razvijalci tudi možnost objave v trgovini App Store.

1.1.2 Opis knjižnice OpenCV

OpenCV (Open Source Computer Vision Library) je knjižnica s programskimi funkcijami, primarno namenjenimi izločanju informacij za nadaljnje odločanje iz videa v realnem času. Knjižnico je razvila družba Intel, danes pa jo podpira Willow Garage in Itseez. Knjižnica je brezplačna pod programsko licenco tipa BSD. Primerna je za uporabo na različnih računalniških in mobilnih platformah [6].

Uradno je knjižnico leta 1999 izdala družba Intel. Na začetku razvoja programske knjižnice OpenCV so bili definirani naslednji cilji:

- 2D in 3D orodja,
- sistem za prepoznavanje obrazov,
- prepoznavanje gest,
- sistem za interakcijo s človekom (angl. human computer interaction),
- mobilna robotika,
- razumevanje gibanja,
- identifikacija objektov,
- segmentacija in prepoznavanje,
- zaznavanje globine iz dveh kamer,
- sistem za prepoznavanje strukture iz gibanja,
- sledenje gibanja,
- navidezno resničnost.

Za podporo nekaterim zgornjim področjem OpenCV vsebuje statistično knjižnico za strojno učenje, ki vključuje:

- metodo za uteževanje učnih primerov glede na njihovo težavnost (angl. boosting),
- učenje s pomočjo odločitvenih dreves,
- gradientna drevesa z metodo za uteževanje učnih primerov (angl. gradient boosting trees),
- algoritem za maksimiranje pričakovanj,
- metodo najbližjih sosedov,
- naivni Bayesov klasifikator,
- umetne nevronske mreže,
- naključne gozdove,
- metodo podpornih vektorjev.

OpenCV je napisan v jeziku C++ in njegov glavni vmesnik (angl. application programming interface) je na voljo v jeziku C++. Še vedno vključuje tudi nepopoln vmesnik (vmesnik, ki ne vključuje vseh funkcionalnosti programske knjižnice OpenCV), ki deluje v programskem jeziku C. Danes obstajajo tudi popolni vmesniki v jeziku Python, Java in Matlab/Octave (od različice 2.5 dalje). Obstaja tudi podpora za druge jezike, kot so C#, Ch in Ruby. Ves nadaljnji razvoj poteka izključno v jeziku C++. Obstaja tudi CUDA vmesnik za delo na grafičnih procesorjih, ki je v razvoju od septembra 2010.

OpenCV deluje na operacijskih sistemih Windows, Android, Maemo, FreeBSD, OpenBSD, iOS, BlackBerry 10, Linux in OS X.

1.2 Cilji

Zaznavanje obraznih izrazov je zahtevno opravilo. Rešitve za zaznavo obrazov na slikah so v veliki večini primerov že vgrajene v operacijske sisteme prenosnih

telefonov (to velja za operacijska sistema iOS in Android) [7, 8]. Rešitve za prepoznavanje obraznih izrazov pa še niso vgrajene v sisteme (ali pa niso dostopne zunanjim razvijalcem). S pomočjo v tem delu razvite knjižnice lahko razvijalec svojemu projektu enostavno priloži knjižnico in uporabi nekaj vnaprej pripravljenih klicev za prepoznavo obraznih izrazov.

Cilji tega dela so:

- razviti knjižnico za prepoznavo obraznih izrazov,
- doseči uporabno klasifikacijsko točnost,
- zagotoviti prenosljivost knjižnice med platformami,
- uporabiti knjižnico na iOS platformi.

Knjižnica mora biti dovolj majhna za uporabo na prenosnih telefonih, enostavna za uporabo in časovno nezahtevna.

Poglavje 2

Predstavitev komercialnih rešitev

Na trgu se pojavljajo različne komercialne rešitve za prepoznavanje obraznih izrazov. Najpogosteje omenjena rešitev je izdelek norveških razvijalcev družbe Noldus [9]. Ta razvijalec je tudi edini, ki podrobneje opisuje delovanje svojega algoritma na spletni strani. Drugi komercialni izdelek, ki je danes dostopen na tržišču, je izdelala družba Percipo [10]. Ta izdelek je med redkimi, ki delujejo na prenosnih telefonih. Na voljo je tudi veliko različnih odprtokodnih rešitev, ki pa so napisane v kodi Matlab in niso na voljo brezplačno (izvorno kodo nekaterih algoritmov lahko prenesemo za neko plačilo) [11], ali pa niso pripravljene za resno uporabo (klasifikator naučen s premajhno bazo) [12].

2.1 Noldus FaceReader 5

Družba Noldus je razvila FaceReader, ki ga oglašujejo kot prvo orodje na svetu, ki je sposobno brez predhodne kalibracije samodejno zaznati človeška čustva.

Družba Noldus opisuje svoj algoritem v treh korakih [13]:

1. V prvem koraku najprej z zelo razširjenim algoritmom Viola-Jones zaznamo človeški obraz na sliki.
2. V drugem koraku izvedemo natančno modeliranje obraza z metodo aktivne predstavitve modela na osnovi videza (angl. active appearance method).

Model vključuje več kot 500 ključnih točk na obrazu in obraznih tekstur med temi točkami. Ključne točke so:

(a) Točke, ki vključujejo obraz (del obraza, ki ga analizira FaceReader).

(b) Točke obraza, ki so enostavno prepoznavne (ustnice, obrvi, nos in oči). Teksture so pomembne, ker nam dajejo dodatne informacije o stanju obraza.

3. V tretjem koraku opravimo klasifikacijo z učenjem umetne nevronske mreže. Za učenje je uporabljenih več kot 10.000 ročno označenih slik.

Aplikacija FaceReader ima shranjenih več modelov. Osnovni model deluje v večini primerov. Obstajajo pa tudi modeli za Vzhodnoazijce, starejše in otroke. Pred analizo obraza je treba izbrati model, ki najbolj ustreza subjektu. Pri nekaterih osebah je treba FaceReader predhodno kalibrirati. Kalibracija je samodejna.

Izhod iz aplikacije FaceReader je na voljo v različnih tabelah. Vsak obrazni izraz (nevtralni, veselje, žalost, jeza, presenečenje, strah in gnus) pa ima dodeljeno vrednost med 0 in 1, ki označuje intenziteto obraznega izraza. Druge aplikacije lahko uporabljajo podatke preko programskega vmesnika (API). Uporabnik lahko spremlja podatke preko grafičnega vmesnika, prikazanega na sliki 2.1.

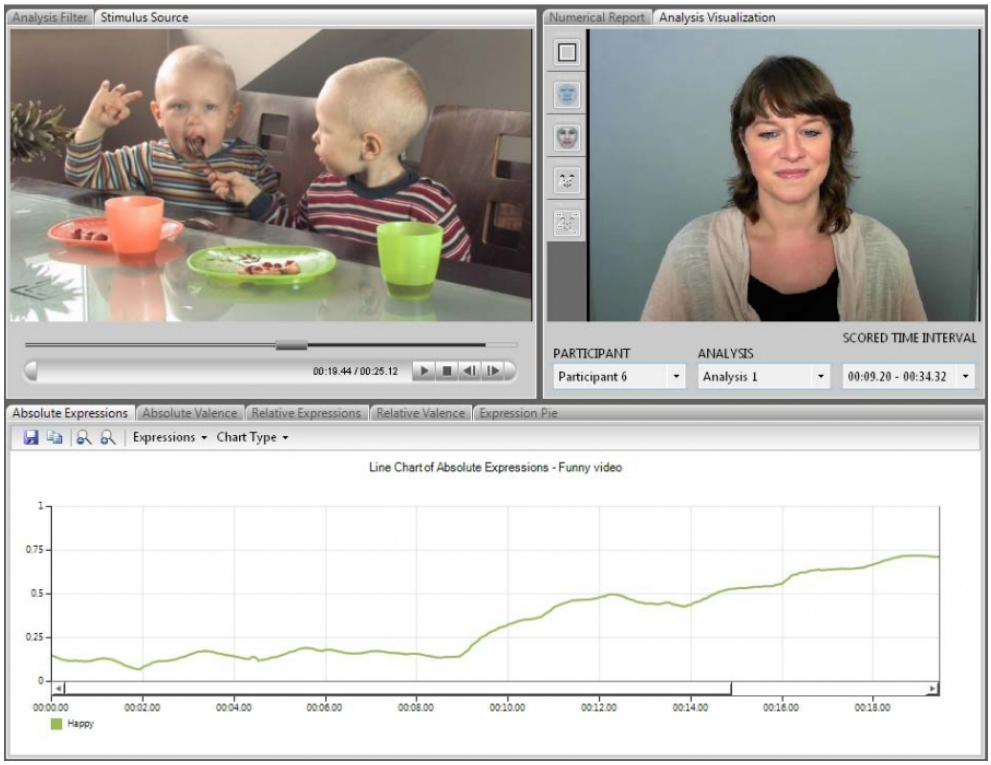
Aplikacija lahko prikazuje stanje obraznih izrazov v realnem času oz., odvisno od računalnika, do 20 slik na sekundo.

Prednosti aplikacije:

- enostavna uporaba,
- predhodna kalibracija ni potrebna,
- možno procesiranje videa v realnem času.

Slabosti:

- za določene osebe je kalibracija obvezna, če želimo dovolj dobro natančnost,
- za dobre rezultate potrebujemo relativno hiter računalnik,
- ne deluje drugje kot na platformi Windows,
- ni primerna za uporabo na prenosnih telefonih.



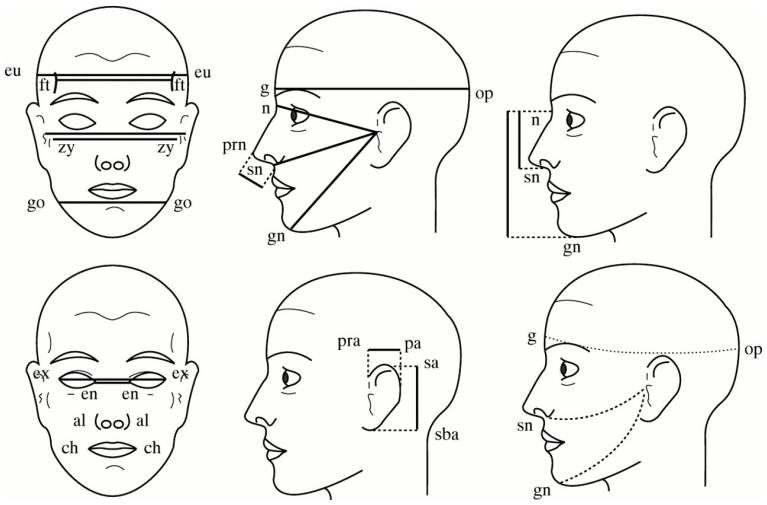
Slika 2.1: Vmesnik programske opreme Noldus

Družba Noldus navaja, da je povprečna klasifikacijska točnost algoritma FaceReader 90 %.

2.2 Percipo

Percipo je razvil rešitev, ki deluje na prenosnih telefonih [10]. Njihov izdelek se od ostalih razlikuje po tem, da deluje na namenskih strežnikih. Odjemalec (prenosni telefon, računalnik ipd.) zato samo posreduje informacije o sliki naprej na strežnik, vsi potrebni izračuni pa se nato izvedejo tam. Ko strežnik zaključi z delom, informacije o obrazu posreduje nazaj na odjemalec. Na ta način se zahtevno preračunavanje ne izvaja na odjemalcih, kar pomeni, da lahko enako dobro napovedujejo obrazne izraze vsi telefoni, tudi najcenejši in najpočasnejši.

Percipo svojega algoritma ne opisuje zelo podrobno. Opisani so le glavni koraki



Slika 2.2: Prikaz ključnih obraznih točk

pred prepoznavanjem obraznih izrazov:

- Iskanje obraza poteka na slikah z velikostjo 42×42 pik.
- Po zaznavi obraza obraz uokvirijo (zaznajo položaj obraza in uredijo obraze, da so vsi v podobni legi) in določijo lego obraza. Lega obraza določa smer gledanja in nagnjenost obraza (slika 2.2).
- Po teh korakih se prične opisovanje obraza [14].

Izhod pri uporabi tehnologije družbe Percipo dobimo s strežnika. Podatki so na voljo za starost, razpoloženje (1 – žalosten, 5 – zelo vesel), spol in točke za lepoto (0 – grd obraz, 100 – lep obraz).

Tehologija deluje le na statičnih slikah, obdelava videa v realnem času pa ni mogoča zaradi povezovanja na strežnik in čakanaja na odgovor strežnika.

Prednosti:

- neodvisno delovanje ne glede na platformo,
- neodvisna hitrost delovanja ne glede na zmogljivost naprave,
- poleg zaznavanja obraznih izrazov nam pove tudi starost in spol osebe.

Slabosti:

- za delovanje potrebujemo internetno povezavo,
- odvisnost od hitrosti povezave,
- ne opiše posameznega obraznega čustva (dobimo le vrednosti od 1–5, 1 – žalosten, 5 – zelo vesel).

Klasifikacijska točnost algoritma družbe Percipo ni navedena v njihovi dokumentaciji ali na predstavitveni strani. Njihova aplikacija *Photo Age* ima zelo slabo oceno in večina komentarjev v spletni trgovini App Store opisuje, da aplikacija ne vrača natančnih rezultatov (napačna starost osebe na sliki) [15].

Poglavje 3

Implementacija

V tem poglavju je opisanih več različnih pristopov za iskanje obrazov na slikah, pretvarjanje podatkov v za računalnik in klasifikatorje primernejšo obliko in različne metode strojnega učenja. Predstavljeni so najpogostejši pristopi in njihove prednosti ter slabosti.

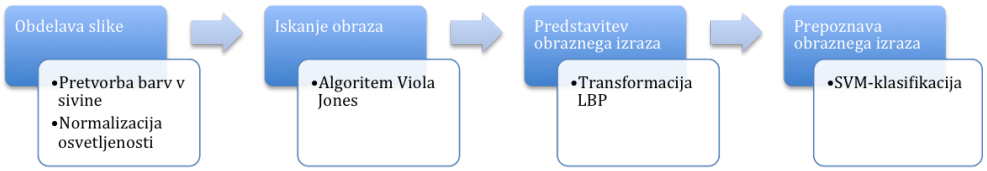
Analiza obraznih izrazov vključuje meritve obraznih premikov in iskanje izraza. Splošni pristop za samodejno iskanje obraznih izrazov vključuje tri korake (slika 3.1): iskanje obraza, luščenje obraznih podatkov in prepoznavanje samega izraza [3].

Iskanje obraza je korak, ki samodejno poišče območje obraza iz vhodne slike ali sekvence. Položaj obraza lahko zaznamo za vsako sliko v sekvenci, ali pa le za prvo in nato sledimo le premikom obraza. Ko je obraz zaznan, je treba izluščiti in predstaviti spremembe obraza, ki jih je povzročil obrazni izraz. Ponavadi se uporabljata dva pristopa: predstavitev z geometričnimi značilkami (angl. geometric features) in predstavitev na osnovi videza (angl. appearance based). Prepoznava obraznega izraza je zadnji korak v algoritmu. Obrazne spremembe so lahko identificirane kot obrazne akcijske enote (angl. facial action units) ali pa kot označeni obrazni izrazi [16].

Slika 3.1: Osnovna struktura sistemov za analizo obraznih izrazov

3.1 Koraki algoritma

Algoritem je sestavljen iz več korakov. Glavni koraki algoritma so prikazani na sliki 3.2:



Slika 3.2: Glavni koraki algortima za prepoznavanje obraznega izraza

- **Obdelava slike:** V tem koraku izvorno sliko (slika, zajeta s telefonsko kamero) pretvorimo v sivinsko sliko. Poleg tega sliko tudi normaliziramo z algoritmom za normalizacijo osvetljenosti.
- **Iskanje obraza:** Na sliki z algoritmom Viola Jones poiščemo vse obraze.
- **Predstavitev obraznega izraza:** Prvi zaznani obraz pretvorimo v za klasifikacijo primernejšo obliko (če je na sliki več obraznih izrazov, jih zavržemo). Združimo najpomembnejše podatke in odstranimo nepomembne. To storimo z LBP-algoritmom.

- **Prepoznava obraznega izraza:** S predhodno naučenim klasifikatorjem (SVM ali k-NN) napovemo obrazni izraz.

3.2 Obdelava slike

Pred iskanjem obraza na sliki je treba sliko obdelati. Poznamo različne načine za obdelavo slike. Pred detekcijo obrazov na sliki pa je še pred uporabo algoritma Viola-Jones, ki je na voljo v knjižnici OpenCV, treba uporabiti algoritem za pretvorbo barvnih slik v sivinske in algoritem za normalizacijo osvetljenosti [21].

Normalizacija osvetljenosti (uravnavanje osvetljenosti s histogrami)

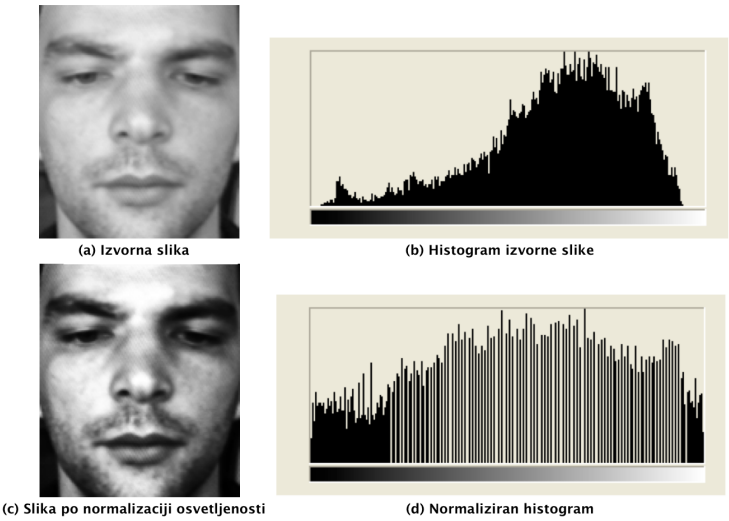
Uravnavanje osvetljenosti s histogrami je metoda prilagajanja kontrasta z uporabo histograma slike [23]. Histogram slike predstavlja relativno frekvenco pojavitev sivih nivojev.

Uravnavanje osvetljenosti s histogramom je operator, ki temelji na posameznih pikah in deluje tako, da na vsaki piki posebej uporabi preslikovalno operacijo neodvisno od sosednjih pik. Ta preslikovalna operacija je kumulativna zgoščevalna funkcija sivinskih nivojev slike. Rezultat preslikovalne funkcije je slika s približno uniformnim histogramom na izravnani sliki.

Kot lahko vidimo na sliki 3.3, je skoraj nemogoče doseči popolnoma izravnan histogram. Poleg tega imajo lahko normalizirane slike več šuma in manj uporabnih podatkov.

Obrazne slike so zelo občutljive na razlike v osvetlitvah, manjše spremembe v svetlobnih pogojih lahko povzročijo velike spremembe v predstavitvi obraza in na ta način otežijo proces iskanja in prepoznave obraznih izrazov. V takšnih primerih je izravnava histograma smiseln korak pred prepoznavanjem obraznih izrazov, saj tako ne poudarimo le kontrasta, ampak tudi robove in podrobnosti obraza.

V nekaterih primerih, kadar je leva stran obraza drugače osvetljena kot desna ali ima drugačno senco, je rezultat z uporabo izravnave osvetljenosti lahko slabši. V takšnih primerih je ta proces manj primeren.



Slika 3.3: Normalizacija osvetljenosti

3.3 Iskanje obraza na sliki

Za iskanje obraza na kompleksnih slikah obstaja veliko različnih metod, ki uporabljajo različne pristope strojnega učenja. Uporabljajo se predvsem nevronske mreže in šibki kaskadni klasifikatorji. Revolucija v prepoznavanju se je zgodila z algoritmom Viola-Jones [18]. Z uporabo šibkih kaskadnih klasifikatorjev na kompleksnih slikah lahko z enostavnimi lastnostmi Haar – po obsežnem učenju algoritma – dobimo zelo dobre rezultate.

OpenCV-knjižnica in naša knjižnica za iskanje obraza na sliki uporabljata klasifikator Viola-Jones.

3.4 Predstavitev obraznih izrazov

Poznamo dva pogosta pristopa preoblikovanja podatkov:

Predstavitev z geometričnimi značilkami (angl. geometric features): Podatki so predstavljeni z oblikami in lokacijami obraznih komponent pridobljenih iz vektorja lastnosti, ki predstavlja geometrijo obraza (slika 3.4). Pomanjkljivost tega pristopa je v tem, da je potrebno samodejno iskanje in sledenje lastnostim obraza. Danes se

v večini primerov uporablja predstavitev na osnovi videza, ki daje enake ali boljše rezultate [22].

Slika 3.4: Nekaj obrazov z geometričnimi značilkami za uporabo pri predstavitvi obraznih izrazov

Predstavitev na osnovi videza (angl. appearance based) podatkov: Podatke predstavimo z različnimi slikovnimi filtri, kot je na primer valčni filter Gabor (angl. Gabor wavelet filter), ki je uporabljen na celotni sliki ali določenem delu slike (slika 3.5). Zaradi večje natančnosti večina raziskovalnega dela temelji na valčnih filtrih Gabor. Ker pa so časovno zahtevni, za mobilne telefone ponavadi niso primerni. Uporabljajo se tudi druge metode s predstavitvijo na osnovi videza, kot so analiza glavnih komponent (angl. principal component analysis), linearna diskriminantna analiza (angl. linear discriminant analysis) in neodvisna analiza komponent (angl. independant component analysis). Dandanes pa se vedno več uporablja tudi nov pristop, ki temelji na lokalnih binarnih vzorcih (angl. local binary paterns). LBP ima nižjo časovno zahtevnost in podobno natančnost kot valčni filtri Gabor [24].

Lokalni binarni vzorci (LBP)

Osnovni LBP-operator so predstavili Ojala et al. [26], gre za način za opisovanje tekstur. Operator označuje pike slike s povprečenjem 3×3 soseščine vsake pike s centralno vrednostjo in vrne rezultat kot binarno število. Bistveni koraki algoritma so prikazani na sliki 3.6:

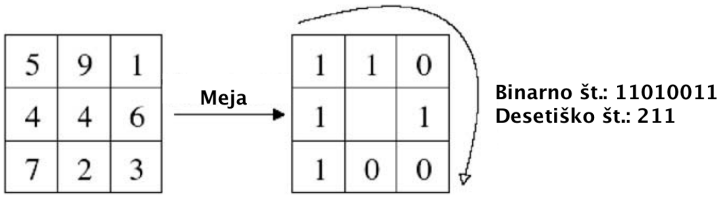
- **Pretvorba vrednosti v binarno število:** Vse vrednosti, ki so manjše od

Slika 3.5: Slike obdelane z uporabo valčnega filtra Gabor

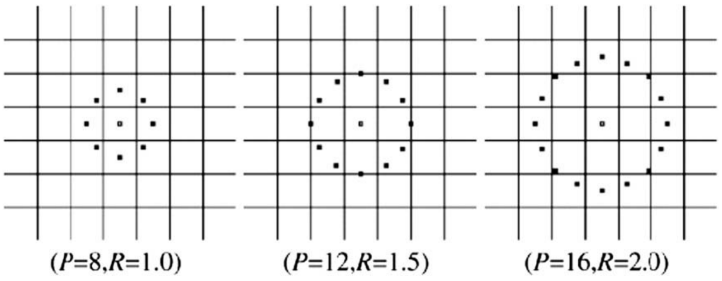
centralne vrednosti (na sliki je to vrednost 4, možne vrednosti so od 0 do 9), dobijo vrednost 0, ostale vrednosti pa vrednost 1.

- **Zapis binarnega števila:** Dobljena binarna števila se prepišejo v eno samo binarno število. Prvo binarno število je v levem zgornjem kotu. Prepisovanje se nadaljuje v smeri urinega kazalca.

Histogram, z 256 koši izračunan preko celotne regije, se uporablja za opis teksture. Izpeljana binarna števila (imenovana lokalni binarni vzorci ali LBP-kode) opisujejo lokalne primitive, vključno z različnimi tipi ukrivljenih robov, prostorov in ravnih območji ter podobne oblike, tako da lahko vsako LBP-kodo interpretiramo kot mikro-tekston. Tekston je manjša osnovna struktura, najdena v naravnih slikah, in šteje za predpogoj človeškega vizualnega dojemanja. Omejitev LBP-operatorja je majhna 3×3 soseska, ki ne more zajeti prevladujočih lastnosti s strukturami, ki zajemajo velika območja slike. Zato je bil operator razširjen za uporabo soseske z različnimi velikostmi. Z uporabo krožne soseske in bilinearno interpolacijo vrednosti dovolimo katerikoli radij in število pik v soseski. Slika 3.7 prikazuje primere razširjenega LBP-operatorja, kjer zapis (P, R) označuje sosesko P enakovredno oddaljenih vzorčnih točk kroga z radijem R, ki označuje krožne simetrične množice sosedov.



Slika 3.6: Osnovni LBP-operator



Slika 3.7: Trije primeri razširjenega LBP-operatorja: krožna (8, 1) soseska, krožna (12, 1,5) soseska in krožna (16, 2) soseska

Uniformni LBP-operator

LBP-operator $\mathrm{LBP}_{\mathrm{P,R}}$ pripravi 2^{p} različnih izhodnih vrednosti, ki ustrezajo 2^{p} različnim binarnim vzorcem, ki se lahko tvorijo iz P pik v množici sosedov. Dokazano je bilo, da nekateri koši vsebujejo več podatkov kot drugi [25]. Zato je mogoče uporabiti samo podmnožico 2^{p} lokalnih binarnih vzorcev za opis tekstur slik. Ojala et al. [26] so poimenovali te ključne vzorce uniformni vzorci. Lokalni binarni vzorec je uniformen, če vsebuje največ dve bitni tranziciji iz 0 v 1 ali iz 1 v 0, ko je binarni niz krožen. 00000000, 001110000 in 11100001 so na primer uniformni vzorci. Uniformni vzorci zavzemajo skoraj 90 % vseh vzorcev v (8, 1) soseski in približno 70 % vseh v soseski (16, 2). Zbiranje teh vzorcev, ki imajo več kot 2 prehoda, v en koš, nam da LBP-operator z oznako $\mathrm{LBP}_{\mathrm{P,R}}{}^{\mathrm{u2}}$, ki ima manj kot 2^{p} različnih vzorcev. Število oznak v soseski 8 pik je na primer 256 za običajni LBP-operator, a samo 59 za $\mathrm{LBP}^{\mathrm{u2}}$-operator [24].

Histogram lokalnih binarnih vzorcev

Po uporabi LBP-operatorja na sliki histogram slike $f_l(x, y)$ lahko definiramo kot

$$H_i = \sum_{x,y} I(f_l(x, y) = i), i = 0, ..., n - 1 \tag{3.1}$$

kjer je n število različnih oznak, ki jih da LBP-operator in

$$I(A) = \begin{cases} 1 & A \text{ je resničen} \\ 0 & A \text{ ni resničen} \end{cases} \tag{3.2}$$

kjer je A je rešitev enačbe $f_l(x, y) = i, i = 0, ..., n - 1$.

LBP-histogram vsebuje informacije o distribucijah lokalnih mikrovzorcev, kot so robovi, pike in ravna območja, na celotnih slikah, tako da jih lahko uporabimo za statističen opis slikovnih karakteristik.

Slike obrazov lahko vidimo kot kompozicije mikrovzorcev, ki jih lahko učinkovito opišemo z LBP-histogrami. Zato je intuitivno, da uporabimo LBP-lastnosti za predstavitev obraznih slik. LBP-histogram, izračunan preko celotne obrazne slike, vsebuje le pojave mikrovzorcev brez predstavitve njihovih lokacij. Če želimo upoštevati tudi informacije o obliki obrazov (algoritem, razvit v temu delu, upošteva tudi to informacijo), lahko obrazne slike razdelimo v majhne regije $R_0, R_1, \ldots, R_M$

pred ekstrakcijo LBP-histogramov. LBP-lastnosti, pridobljene iz vsake podregije, nato združimo v prostorsko obogaten histogram lastnosti, ki je definiran kot

$$H_{i,j} = \sum_{x,y} I\{f_l(x,y) = i\} I\{(x,y) \in R_j\} \tag{3.3}$$

kjer je i = 0,...n-1, j = 0,...m-1 [24].

Pridobljeni histogram predstavlja lokalno teksturo in globalno obliko obraznih slik. Nekatere parametre lahko optimiziramo za boljšo prepoznavo lastnosti. Prilagodimo lahko LBP-operator ali število območji. V tem delu smo uporabili LBP-operator z 59 koši $\mathrm{LBP}_{8,2}{}^{\mathrm{u2}}$ ter razdelili sliko obraza v 7×7 enako velikih območij. To pomeni, da smo sliko razdelili v 49 enako velikih območij in jo predstavili z LBP-histogrami dolžine 2891 (59×49).

3.5 Prepoznava obraznega izraza

Za prepoznavanje obraznega izraza obstaja veliko različnih tehnik, kot so na primer nevronske mreže, metoda podpornih vektorjev, metoda najbližjih sosedov, Bayesova mreža in druge metode strojnega učenja. V našem algoritmu smo za učenje in testiranje uporabili metodo najbližjih sosedov in metodo podpornih vektorjev, končna rešitev pa uporablja za prepoznavanje obraznih izrazov metodo podpornih vektorjev.

3.5.1 Metoda najbližjih sosedov (k-NN)

Metoda k-NN je neparametrična metoda, ki se uporablja za klasifikacijo in regresijo [27]. Metoda je neparametrična, ker ne predpostavlja nikakršne predhodne podatkovne distribucije. V obeh primerih se vhodni podatki sestavijo iz k najbližjih učnih primerov v izbranem prostoru. Izhod je odvisen od tega, ali je k-NN uporabljen za klasifikacijo ali regresijo:

- V k-NN-klasifikaciji je izhod odvisen od pripadnosti razredu. Objekt je klasificiran z večinskim glasom njegovih sosedov. Če je k=1, potem je razred dodeljen glede na enega najbližjega soseda.

- V k-NN-regresiji je izhod povprečna vrednost vrednosti njegovih k najbližjih sosedov.

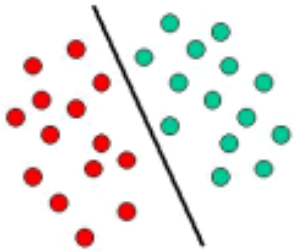
K-NN je eden izmed najbolj preprostih algoritmov od vseh algoritmov strojnega učenja. Sosede vzame iz množice objektov, za katere je razred znan. Temu lahko rečemo tudi učna množica algoritma, a predhodno učenje ni potrebno. Slabost algoritma je, da je zelo občutljiv na lokalno strukturo podatkov.

Algoritem: Učni primeri so vektorji v večdimenzionalnem prostoru, vsak vektor ima označen razred. V učnem koraku se le shranijo vektorji ter oznake razredov iz učne množice [35]. V koraku klasifikacije je k konstanta, ki jo določi uporabnik. Neoznačen vektor (ali testna množica) je klasificiran tako, da se določi oznaka razreda, ki se največkrat pojavi v množici k učnih primerov v odvisnosti od neoznačenega vektorja. Ponavadi se za merjenje razdalje uporablja evklidska razdalja (OpenCV knjižnica in naš algoritem uporabljata evklidsko razdaljo). Pogosto se lahko klasifikacijsko natančnost algoritma k-NN izboljša tudi, če je mera razdalje določena s posebnimi algoritmi, kot je analiza komponent sosedov [29]. Natančnost k-NN-algoritma je zelo zmanjšana, če so v podatkih šumni ali nepomembni atributi.

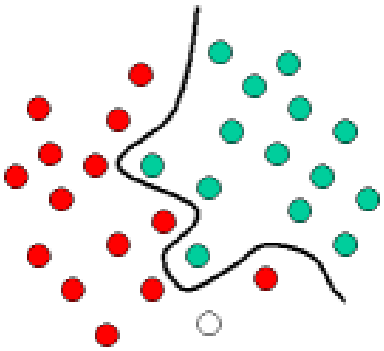
Izbira parametra k: Parameter k je v večini primerov odvisen od podatkov; splošno večje vrednosti parametra k preprečujejo šum pri klasifikacijah, a zameglijo meje med razredi. Dober parameter k lahko določimo s prečnim preverjanjem in različnimi hevrističnimi metodami, npr. s hiperparametrično optimizacijo [30] in metodo prečnega preverjanja [31] (parameter k za algoritme razvite v tem delu je bil določen s hiperparametrično optimizacijo).

3.5.2 Metoda podpornih vektorjev (SVM)

Metode podpornih vektorjev temeljijo na konceptu odločitvenih ploskev, ki določajo odločitvene meje. Odločitvena ploskev je tista ploskev, ki ločuje množico različnih razredov [36]. Shematični primer je prikazan na sliki 3.8. Na tem primeru je vidno, da objekti pripadajo zeleni ali rdeči množico. Ločitvena črta prikazuje mejo med levo stranjo, kjer so vsi zeleni objekti, in desno stranjo, kjer so vsi rdeči objekti. Vsak nov primer (bela pika) je nato klasificiran kot rdeč ali zelen, odvisno od tega, na katero stran meje sodi.



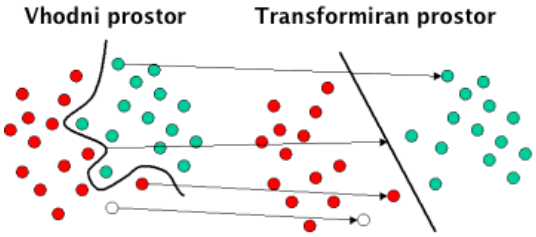
Slika 3.8: Optimalna ločitev razredov



Slika 3.9: Primer, ko enostavna črta ne ločuje razredov dovolj dobro

Primer iz slike 3.8 je klasičen primer linearne klasifikacije, torej klasifikacije, ki ločuje množico primerov (v tem primeru zelene in rdeče objekte) z ravno črto. Večina odločitvenih primerov pa ni tako enostavnih in za njih potrebujemo veliko bolj kompleksne strukture, da naredimo optimalno ločitev, torej da pravilno uvrstimo objekte v pravilne razrede. Slika 3.9 prikazuje primer, ko enostavna črta ne zadostuje za optimalno ločitev primerov v ločene razrede. Odločitvena opravila, ki ločujejo s pomočjo ločitvenih črt za razločevanje primerov iz različnih razredov, se imenujejo hiperploskovni klasifikatorji. Metode podpornih vektorjev so primerne za taka opravila.

Slika 3.10 prikazuje glavno idejo za metodo podpornih vektorjev. Tu vidimo prvotno razporeditev objektov (leva stran sheme), ki je bila prerazporejena z množico matematičnih jedrnih (angl. kernel) funkcij. Ta proces se imenuje tudi mapiranje (transformacija). V tej novi prerazporeditvi objektov (desna stran slike)



Slika 3.10: Primer razporejanja objektov z matematičnimi funkcijami

so objekti linearno ločljivi, zato nam ni treba izračunavati kompleksne krivulje, ampak uporabimo enostavno linearno enačbo, ki ločuje med zelenimi in rdečimi objekti. Če dimenzijo dovolj povečamo, postanejo vsi razredi vektorjev linearno ločljivi. Pri tem nastaneta dve težavi:

- Transformacija vektorjev v prostor z višjo dimenzijo je računsko zahtevna operacija.
- Inverzna transformacija hiperravnino v prostoru z višjo dimenzijo spremeni v zelo zapleteno hiperploskev v prostoru z nižjo dimenzijo.

Če za transformacijo uporabimo posebne funkcije, ki se jim reče jedrne funkcije, lahko te težave omilimo. Transformacijo in inverzno transformacijo lahko izvedemo, ne da bi jo dejansko računsko izvedli. Tudi tukaj za popoln opis meje med razredi zadošča uporaba samo enega dela učnih vektorjev (podporni vektorji).

Različice metode podpornih vektorjev

Za izgraditev optimalnega hiperprostora SVM uporablja iterativni učni algoritem, ki se uporablja za minimiranje funkcije napake. Odvisno od oblike funkcije lahko SVM modele razporedimo v štiri različne skupine [36]:

- klasifikatorski SVM tipa 1 (znan tudi kot C-SVM),
- klasifikatorski SVM tipa 2 (znan tudi kot nu-SVM),
- regresijski SVM tipa 1 (znan tudi kot regresijski epsilon-SVM),
- regresijski SVM tipa 2 (znan tudi kot regresijski nu-SVM).

Algoritmi, razviti v tem delu, za učenje in klasifikacijo uporabljajo SVM tipa 1.

Jedrne funkcije

Obstaja več različnih jedrnih funkcij, ki jih lahko uporabimo v metodi podpornih vektorjev [32]:

- linearna,
- polinomska,
- radialna bazna funkcija (RBF),
- sigmoidna.

Algoritmi, razviti v tem delu, za učenje in klasifikacijo uporabljajo linearno jedrno funkcijo in radialno bazno jedrno funkcijo.

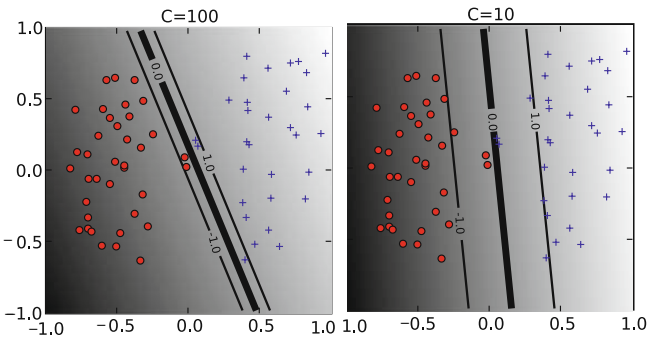
Parametri

Metode podpornih vektorjev z različnimi jedrnimi funkcijami uporabljajo parametre, s katerimi lahko prilagajamo natančnost klasifikatorja, omilimo preveliko prilagajanje podatkom in pospešimo izvajanje celotnega algoritma [37]. Nekaj najpogosteje uporabljenih parametrov:

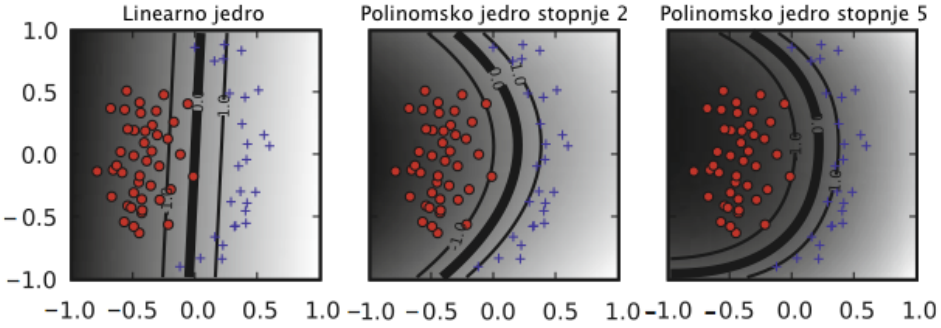
Konstanta mehkega roba C: Konstanta C vpliva na meje odločanja. Večja vrednost konstante C bolj kaznuje napake. To je vidno na sliki 3.11, kjer dve točki blizu hiperploskve vplivata na orientacijo in ustvarita hiperploskev, ki se približa številnim drugim podatkovnim točkam. Manjša vrednost C (desna stran slike 3.11) omogoča ignoriranje točk blizu meje in poveča rob. Orientacija hiperploskve v tem primeru se spremeni in ustvari veliko večji rob za druge podatke.

Stopnja polinomskega jedra: Stopnja polinomskega jedra vpliva na prilagodljivost nastalega klasifikatorja. Najmanjša stopnja polinomskega jedra je linearno jedro, ki ponavadi ni dovolj dobro za reševanje nelinearnih odvisnosti. Na sliki 3.12 lahko vidimo, da je že stopnja polinomskega jedra 2 dovolj prilagodljiva za ločevanje med dvema razredoma z dovolj velikim robom. Stopnja 5 ustvari podobne meje, a z večjimi zavoji.

Vrednost parametra γ: Izračunana vrednost Gaussove ali RBF-jedrne funkcije je zelo majhna. Parameter γ zato močno vpliva na končno vrednost te funkcije. Kadar je γ majhna (zgornji levi del na sliki 3.13), je odločitvena meja skoraj

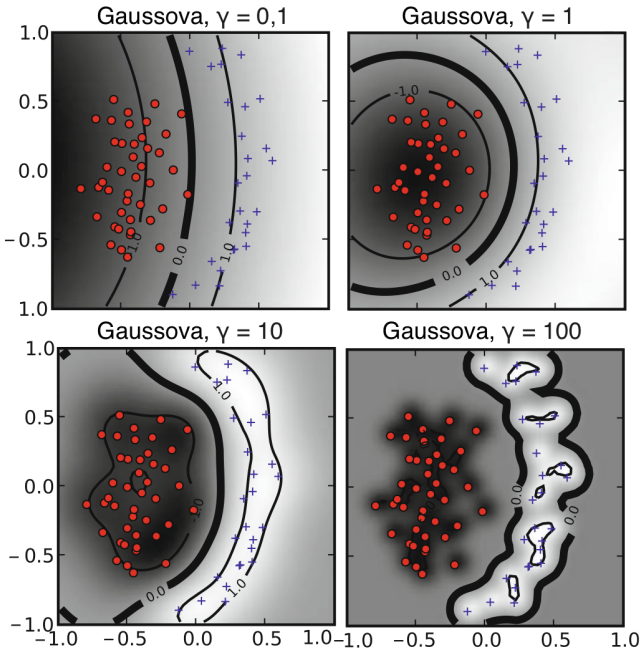


Slika 3.11: Vpliv parametra C na orientacijo hiperploskve



Slika 3.12: Vpliv stopnje polinomskega jedra na ločevanje razredov

linearna. Ko povečujemo γ, se fleksibilnost odločitvene meje povečuje. Velike vrednosti γ lahko pripeljejo do prevelikega prilagajanja.



Slika 3.13: Vpliv parametra γ na fleksibilnost odločitvene meje

Poglavje 4

Rezultati

Algoritmi so bili naučeni in testirani na dveh različnih podatkovnih bazah. Zaradi konsistentnosti ocenjevanja smo na obeh podatkovnih bazah uporabili enako metodo za ocenjevanje točnosti. Podatkovno bazo smo enkratno razbili na učno in testno množico. Prečnega preverjanja nismo uporabili zaradi prevelike podatkovne baze Kaggle. Učenje in testiranje bi trajalo preveč časa. Pred razbitjem na dve množici smo razrede v podatkovni bazi Cohn-Kanade prerazporedili, tako da so bili enakomerno porazdeljeni po celotni učni in testni množici (angl. stratified holdout) [33], ker je baza v primerjavi s podatkovno bazo Kaggle veliko manjša in bi se lahko zgodilo, da nekateri primeri razredov ne bi bili zastopani v učni ali testni množici [34]. Pri obeh bazah je del za testiranje zajemal 10 odstotkov celotne baze in ni bil nikoli uporabljen za učenje algoritmov. Slike, na katerih obraz ni bil prepoznan, niso bile uporabljene niti za učenje niti za testiranje. Učenje in testiranje je potekalo na računalniku z 2,0 GHz štiri-jedrnim Intel Core i7 procesorjem in operacijskim sistemom OS X (10.9). Za primerjanje rezultatov smo uporabili klasifikacijsko točnost. Klasifikacijska točnost [35] je definirana z enačbo 4.1:

$$T = \frac{N^{(p)}}{N} \times 100\% \tag{4.1}$$

N – število vseh možnih primerov problemov na danem področju

$N^{(p)}$ – število pravilnih rešitev primerov

Najnižjo še sprejemljivo klasifikacijsko točnost je mogoče enostavno doseči s klasifikacijo vseh primerov v večinski razred [35]. Klasifikacijska točnost večinskega

razreda je definirana z enačbo 4.2:

$$Acc_m = max_C \left\{ \frac{n(C)}{n} \right\} \times 100\% \tag{4.2}$$

$n(C)$ – število učnih primerov iz razreda C

n – število vseh učnih primerov

Poleg tega so rezultati zapisani tudi v matrikah pravilnih in napačnih razvrstitev (angl. confussion matrix), ki nam prikazuje klasifikacijsko točnost posameznega razreda. Pravilne napovedi so zapisane v diagonali matrike. Vse ostale napovedi so napačne. Posamezen element matrike je razmerje med številom napovedanih za izbran razred in številom vseh v tem razredu. Elementi posamezne vrstice se seštejejo v 100 %.

4.1 Učni podatki

Za učenje sta bili uporabljeni dve učni množici. Množica iz podatkovne baze obrazov Cohn-Kanade [38] in iz tekmovanja Kaggle [39].

4.1.1 Podatkovna baza Kaggle

Ta baza je bila narejena za potrebe tekmovanja z naslovom *"Challenges in Representation Learning: Facial Expression Recognition Challenge"*. Vsebuje sivinske slike v velikosti 48×48 pik (slika 4.1) z označenimi obraznimi izrazi. Obrazi na slikah so bili samodejno zaznani in razporejeni tako, da vedno zavzemajo enako površino na sliki ter so bolj ali manj na sredini. V množici se nahajajo tudi slike risanih junakov in otrok, kar otežuje klasifikacijo in testiranje. V bazi so slike za 7 različnih obraznih izrazov skupaj z nevtralnim izrazom (ostali izrazi: jeza, gnus, strah, veselje, žalost in presenečenje). Za učenje algoritma je bilo uporabljenih skupno 28.709 slik različnih subjektov. 3.189 subjektov je bilo uporabljenih za testno množico.

Slika 4.1: Primer slik iz podatkovne baze Kaggle

4.1.2 Podatkovna baza Cohn-Kanade

Ta baza je namenjena za raziskovanje samodejne obrazne analize in sinteze ter za druge študije (slika 4.2). Slike v bazi Cohn-Kanade vsebujejo sekvence slik za 123 različnih subjektov. Sekvence vsebujejo sliko subjekta z nevtralnim izrazom, ki se nato stopnjuje do želenega obraznega izraza. Za naš algoritem je bila uporabljena različica baze 2, ki se imenuje tudi CK+ in vključuje tako zaigrane kot tudi spontane obrazne izraze in dodatne metapodatke. Poleg tega baza vključuje tudi označene obrazne izraze, kar je za učenje algoritma ključnega pomena. Slike v tej bazi so vedno enako osvetljene, enako velike (640×490 ali 640×480 pik) in sivinske. V bazi so slike za 8 različnih obraznih izrazov skupaj z nevtralnim izrazom (ostali izrazi: jeza, gnus, prezir, strah, veselje, žalost in presenečenje). Vsi subjekti niso imeli slik za vse obrazne izraze. Za učenje in testiranje algoritma je bilo uporabljenih skupno 1.308 slik 118 različnih subjektov, vsak obrazni izraz pa

je bil predstavljen s 3 različnimi slikami na subjekt. Množica za učenje je vsebovala 1176 slik, množica za testiranje pa 132 slik. 5 subjektov ni bilo vključenih v fazo učenja in testiranja algoritma, ker ni bilo zapisov za obrazni izraz na posameznih slikah.

Slika 4.2: Primer slik iz podatkovne baze Cohn-Kanade

4.2 Učenje z metodo najbližjih sosedov (k-NN)

V tem razdelku so zapisani rezultati učenja z metodo najbližjih sosedov. Klasifikator smo učili in testirali na neobdelanih podatkih in podatkih, obdelanih z metodo LBP. Parameter k smo določili, tako da smo množico razdelili na učno in testno ter nato primerjali klasifikacijsko točnost za različne vrednosti parametra k. Izbrana je bila vrednost parametra k, ki je dosegla najvišjo klasifikacijsko točnost.

4.2.1 Neobdelani podatki

Testirali smo k-NN-klasifikator brez predhodne obdelave podatkov z vrednostjo parametra k, nastavljeno na 10. Klasifikacijska točnost algoritma, ocenjenega na neodvisni testni množici, je 33,3 %, kar je še vedno več od točnosti večinskega klasifikatorja, ki je 25,0 %. Matrika pravilnih in napačnih klasifikacij v tabeli 4.1 prikazuje, da je algoritem v večini primerov napovedal nevtralni izraz. Razlog za majhno klasifikacijsko točnost je, da je večina parametrov (atributov), ki opisujejo obrazni izraz, nepomembnih [35] (preoblikovanje podatkov z metodo LBP je namenjeno tudi odstranitvi nepomembnih atributov). Poleg nizke točnosti klasifikatorja je ta rešitev neprimerna za uporabo na prenosnem telefonu zaradi velikosti samega klasifikatorja (shranjena učna množica je velika 165,4 MB) in počasne klasifikacije (41 primerov na sekundo). k-NN-klasifikator z enakimi nastavitvami, naučen in testiran s podatkovno bazo Kaggle se je izkazal bolje. Klasifikacijska točnost testiranega algoritma je večja (41,9 %) od k-NN-klasifikatorja Cohn-Kanade (je tudi mnogo večja od točnosti večinskega klasifikatorja, ki je pri podatkovni bazi Kaggle 27,9 %). Tudi v tabeli pravilnih in napačnih klasifikacij 4.2 lahko vidimo enakomernejšo porazdelitev napovedi (vsak obrazni izraz je bil napovedan vsaj enkrat). Razlog za boljšo klasifikacijsko točnost je najverjetneje veliko večja učna množica. Je pa klasifikator mnogo večji (1.804 MB) in zelo počasen (4 primeri na sekundo) ter zato tudi neprimeren za uporabo na prenosnih telefonih.

	nevtralni %	jeza %	prezir %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**69.7**	9.1	6.1	0	0	9.1	6.1	0
jeza	80.0	**0**	0	0	0	0	20.0	0
prezir	0	100.0	**0**	0	0	0	0	0
gnus	50.0	5.6	5.6	**5.6**	0	22.2	11.1	0
strah	50.0	50.0	0	0	**0**	0	0	0
veselje	47.6	0	0	19.0	0	**33.3**	0	0
žalost	66.7	22.2	11.1	0	0	0	**0**	0
presenečenje	33.3	3.7	0	14.8	0	0	0	**48.1**

Tabela 4.1: Matrika pravilnih in napačnih razvrstitev (k-NN, podatkovna baza Cohn-Kanade, neobdelani podatki, klasifikacijska točnost 33,3 %).

	nevtralni %	jeza %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**53.3**	8.8	0	5.4	22.6	5.6	4.4
jeza	26.6	**28.8**	1.1	8.3	22.7	6.8	5.8
gnus	20.7	6.9	**10.3**	6.9	37.9	10.3	6.9
strah	28.6	13.4	0.4	**20.1**	22.3	5.7	9.5
veselje	15.0	6.7	0.5	3.2	**68.6**	3.2	2.7
žalost	35.0	16.7	1.6	6.0	24.9	**11.7**	4.1
presenečenje	22.1	10.0	1.1	6.8	12.9	5.0	**42.1**

Tabela 4.2: Matrika pravilnih in napačnih razvrstitev (k-NN, podatkovna baza Kaggle, neobdelani podatki, klasifikacijska točnost 41,9 %).

4.2.2 Podatki, obdelani z metodo LBP

Predhodna obdelava podatkov zmanjša število nepotrebnih atributov in šum v podatkih. V tem razdelku smo testirali k-NN-klasifikator. Klasifikator je bil naučen s podatkovno bazo Cohn-Kanade ali Kaggle. Podatki so bili obdelanimi z metodo LBP. Vrednost parametra k je enaka kot pri prejšnjih testih.

Klasifikacijska točnost algoritma, ocenjenega na neodvisni testni množici podatkovne baze Cohn-Kanade, je 37,9 %, kar je več kot pri klasifikatorju Cohn-Kanade, naučenem z neobdelanimi podatki. Iz matrike pravilnih in napačnih klasifikacij v tabeli 4.3 lahko razberemo, da je klasifikator v primerjavi s klasifikatorjem, naučenim z neobdelanimi podatki, pravilno napovedal tudi obrazni izraz prezir, ostale pa je vse (razen presenečenja) napovedal z večjo ali enako točnostjo. Razlog za večjo točnost je v obdelavi podatkov, saj smo tako odstranili večino nepomembnih podatkov in s tem zmanjšali količino (dimenzionalnost) podatkov [24]. Velikost samega klasifikatorja (49,8 MB) je tudi veliko manjša od klasifikatorja, naučenega z neobdelanimi podatki, posledično je zmanjšana tudi hitrost samega napovedovanja (660 primerov na sekundo).

Če primerjamo k-NN-klasifikator z enakimi parametri kot v zgornjem primeru, le da za učenje in testiranje uporabimo podatkovno bazo Kaggle, dobimo klasifikacijsko točnost 44,9 %. Iz matrike pravilnih in napačnih klasifikacij v tabeli 4.4 razberemo, da se je klasifikator izkazal bolje od klasfikatorja, ki je bil naučen z neobdelanimi podatki. Pravzaprav se je ta klasifikator odrezal najbolje od vseh te-

	nevtralni %	jeza %	prezir %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**84.8**	6.1	3.0	3.0	3.0	0	0	0
jeza	80.0	**0**	0	0	13.3	0	6.7	0
prezir	66.7	0	**33.3**	0	0	0	0	0
gnus	61.1	0	0	**22.2**	0	16.7	0	0
strah	100.0	0	0	0	**0**	0	0	0
veselje	47.6	0	4.8	4.8	0	**42.9**	0	0
žalost	100.0	0	0	0	0	0	**0**	0
presenečenje	48.1	3.7	0	11.1	7.4	0	0	**29.6**

Tabela 4.3: Matrika pravilnih in napačnih razvrstitev (k-NN, Cohn-Kanade, podatki obdelani z LBP, klasifikacijska točnost 37,9 %).

stiranih k-NN-klasifikatorjev. Velikost samega klasifikatorja (453,1 MB) je manjša od klasifikatorja, naučenega z neobdelanimi podatki, zaradi česar je manjša tudi hitrost samega napovedovanja (28 primerov na sekundo).

	nevtralni %	jeza %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**46.5**	6.6	1.5	8.3	16.8	7.8	12.7
jeza	21.9	**34.2**	2.2	6.1	16.2	12.2	7.2
gnus	6.9	34.5	**20.7**	6.9	17.2	6.9	6.9
strah	21.6	13.8	1.1	**17.7**	20.5	7.8	17.7
veselje	11.3	5.6	0.3	2.2	**72.4**	2.7	5.5
žalost	28.7	12.3	1.6	8.2	20.2	**20.5**	8.5
presenečenje	15.7	7.1	0.7	10.4	11.1	1.1	**53.9**

Tabela 4.4: Matrika pravilnih in napačnih razvrstitev (k-NN, podatkovna baza Kaggle, podatki obdelani z LBP, klasifikacijska točnost 44,9 %).

4.3 Učenje z metodo podpornih vektorjev (SVM)

V tem razdelku so zapisani rezultati učenja z metodo podpornih vektorjev, ki je zahtevnejša in težje razumljiva metoda strojnega učenja (v primerjavi z metodo

najbližjih sosedov). Klasifikator smo učili in testirali na neobdelanih podatkih in podatkih obdelanih z metodo LBP (z nastavitvami zapisanimi v tabelah 4.5 in 4.6). Optimalne nastavitve parametrov so bile določene za vsak algoritem posebej z metodo (`CvSVM::train_auto`), ki je na voljo v OpenCV knjižnici.

SVM jedro	C	Stopnja polinomskega jedra	γ
neobdelani podatki			
linearno	$1,0 \times 10^{-1}$	1	/
RBF	$1,0 \times 10^{-1}$	/	$1,0 \times 10^{-5}$
podatki obdelani z LBP			
linearno	$1,25 \times 10^{1}$	1	/
RBF	$3,125 \times 10^{2}$	/	$3,375 \times 10^{-2}$

Tabela 4.5: Nastavitve SVM-klasifikatorjev, podatkovna baza Cohn-Kanade

SVM jedro	C	Stopnja polinomskega jedra	γ
neobdelani podatki			
linearno	$5,0 \times 10^{-1}$	1	/
RBF	$5,0 \times 10^{-1}$	/	$3,0 \times 10^{-5}$
podatki obdelani z LBP			
linearno	$1,0 \times 10^{-1}$	1	/
RBF	$2,5$	/	$5,0 \times 10^{-1}$

Tabela 4.6: Nastavitve SVM klasifikatorjev, Kaggle podatkovna baza

4.3.1 Neobdelani podatki

Testirali smo SVM klasifikatorje, ki smo jih naučili s podatki brez predhodne obdelave. SVM klasifikatorji so uporabljali linearno ali RBF jedro.

Linearno jedro Klasifikacijska točnost algoritma naučenega s Cohn-Kanade podatkovno bazo in ocenjenega na neodvisni testni množici je 70,4 %. Iz matrike pravilnih in napačnih klasifikacij v tabeli 4.7 lahko vidimo, da je imel klasifikator težave z obraznimi izrazi prezir, strah in žalost. Teh primerov je v Cohn-Kanade bazi manj kot ostalih obraznih izrazov. Naučeni klasifikator je veliko bolj primeren za uporabo na prenosnih telefonih, saj je veliko manjši (12,3 MB) od celotne učne

množice potrebne za napovedovanje pri k-NN metodi (165,4 MB) in tudi veliko hitrejši (2.252 primerov na sekundo).

	nevtralni %	jeza %	prezir %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**93.9**	6.1	0	0	0	0	0	0
jeza	0	**93.3**	0	6.7	0	0	0	0
prezir	100.0	0	**0**	0	0	0	0	0
gnus	16.7	16.7	0	**66.7**	0	0	0	0
strah	0	50.0	50.0	0	**0**	0	0	0
veselje	28.6	0	0	14.3	0	**57.1**	0	0
žalost	66.7	33.3	0	0	0	0	**0**	0
presenečenje	11.1	0	0	3.7	0	0	0	**85.2**

Tabela 4.7: Matrika pravilnih in napačnih razvrstitev (SVM z linearnim jedrom, Cohn-Kanade, neobelani podatki, klasifikacijska točnost 70,4 %).

Klasifikacijska točnost algoritma naučenega s Kaggle podatkovno bazo in ocenjenega na neodvisni testni množici je 27,4 %. Poleg tega, da se je klasifikator odrezal slabše kot večinski klasifikator (27,9 % – obrazni izraz veselje), je tudi iz matrike pravilnih in napačnih klasifikacij v tabeli 4.8 razvidno, da se je klasifikator odrezal veliko slabše od vseh k-NN klasifikatorjev. Največjo klasifikacijsko točnost je klasifikator dosegel pri obraznem izrazu veselje (52,0 %), ki je v tej podatkovni bazi tudi najpogosteje zastopan. Velikost samega klasifikatorja (9,2 MB) je dovolj majhna za uporabo na prenosnem telefonu, prav tako tudi hitrost klasifikacije (3.165 primerov na sekundo).

RBF-jedro Klasifikator naučen s podatkovno bazo Cohn-Kanade in RBF-jedrom se je odrezal najslabše od vseh testiranih klasifikatorjev. Klasifikacijska točnost algoritma je zgolj 25,0 %. V matriki pravilnih in napačnih klasifikacij (tabela 4.9) lahko vidimo, da je algoritem napovedal le en obrazni izraz. To je privzeti nevtralni obrazni izraz. Klasifikator je velik 174,5 MB, hitrost klasifikacije pa je 39 primerov na sekundo.

Enako slabo se je odrezal tudi klasifikator naučen s Kaggle podatkovno bazo. Dosegel je enako klasifikacijsko točnost 25,0 %, matrika pravilnih in napačnih klasifikacij (tabela 4.9) pa prikazuje enake rezultate kot pri SVM z RBF jedrom in naučenim s Cohn-Kanade podatkovno bazo (stolpca z rezultati za obrazni izraz

	nevtralni %	jeza %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**11.4**	20.0	1.0	10.7	24.8	23.8	8.3
jeza	6.1	**21.6**	0.4	12.6	23.7	27.7	7.9
gnus	3.4	17.2	**0**	10.3	51.7	17.2	0
strah	8.8	14.8	1.1	**17.3**	23.7	24.4	9.9
veselje	4.1	13.0	0.3	6.5	**52.0**	17.7	6.3
žalost	8.5	15.5	0.3	13.9	22.7	**26.5**	12.6
presenečenje	10.7	9.3	0.7	16.1	16.8	27.1	**19.3**

Tabela 4.8: Matrika pravilnih in napačnih razvrstitev (SVM z linearnim jedrom, Kaggle, neobelani podatki, klasifikacijska točnost 27,4 %).

	nevtralni %	jeza %	prezir %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**100.0**	0	0	0	0	0	0	0
jeza	100.0	**0**	0	0	0	0	0	0
prezir	100.0	0	**0**	0	0	0	0	0
gnus	100.0	0	0	**0**	0	0	0	0
strah	100.0	0	0	0	**0**	0	0	0
veselje	100.0	0	0	0	0	**0**	0	0
žalost	100.0	0	0	0	0	0	**0**	0
presenečenje	100.0	0	0	0	0	0	0	**0**

Tabela 4.9: Matrika pravilnih in napačnih razvrstitev (SVM z RBF-jedrom, Cohn-Kanade in Kaggle, neobelani podatki, klasifikacijska točnost 25,0 %).

prezir v tem primeru ni). Velikost klasifikatorja je 205,5 MB, hitrost klasifikacije pa je 23 primerov na sekundo.

Slabe rezultate SVM-klasifikatorjev z RBF-jedrom lahko pripišemo slabo izbranim parametrom ali slabo pripravljenim podatkom. RBF-jedro je bolj občutljivo na izbrane parametre kot linearno jedro. Poleg tega je RBF-jedro tudi veliko bolj občutljivo na nepomebne atribute kot linearno jedro, kar pomeni, da lahko s preblikovanjem podatkov izboljšamo rezultate [41].

4.3.2 Podatki, obdelani z metodo LBP

V tem razdelku smo testirali SVM-klasifikatorje, naučene s predhodno obdelanimi podatki z uporabo algoritma LBP. SVM-klasifikatorji so uporabljali linearno ali RBF-jedro.

Linearno jedro Klasifikacijska točnost algoritma, naučenega s podatkovno bazo Cohn-Kanade in ocenjenega na neodvisni testni množici, je 87,1 %. Tudi v matriki pravilnih in napačnih klasifikacij v tabeli 4.10 lahko vidimo zelo enakomerno razporeditev klasifikacijskih točnosti. Slabše se je odrezal le pri preziru in strahu. Naučeni klasifikator je zelo primeren za uporabo na prenosnih telefonih, saj je zelo majhen (1,6 MB) in zelo hiter (16.154 primerov na sekundo).

	nevtralni %	jeza %	prezir %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**100.0**	0	0	0	0	0	0	0
jeza	0	**86.7**	0	0	0	0	13.3	0
prezir	66.7	0	**33.3**	0	0	0	0	0
gnus	0	0	0	**100.0**	0	0	0	0
strah	50.0	0	0	0	**50.0**	0	0	0
veselje	0	0	0	4.8	0	**95.2**	0	0
žalost	11.1	11.1	0	0	0	0	**66.7**	11.1
presenečenje	18.5	0	0	3.7	0	0	0	**77.8**

Tabela 4.10: Matrika pravilnih in napačnih razvrstitev (SVM z linearnim jedrom, Cohn-Kanade, podatki obdelani z LBP, klasifikacijska točnost 87,1 %).

SVM-klasifikator, naučen s podatkovno bazo Kaggle, ima veliko slabšo klasifikacijsko točnost od SVM-klasifikatorja, naučenega s podatkovno bazo Cohn-Kanade, a veliko boljšo točnost od klasifikatorja, naučenega s podatkovno bazo Kaggle pri uporabi RBF-jedra in neobdelanih podatkih. Klasifikacijska točnost je 50,1 %, v matriki pravilnih in napačnih razvrstitev, prikazani v tabeli 4.11, pa lahko vidimo, da je klasifikator z majhno klasifikacijsko točnostjo napovedoval strah in žalost, medtem ko gnusa ni nikoli napovedal pravilno. Velikost samega klasifikatorja je manjša (1,2 MB) od SVM-klasifikatorja, naučenega s podatkovno bazo Cohn-Kanade (1,6 MB). Hitrost napovedovanja tega klasifikatorja je 21.411

primerov na sekundo.

	nevtralni %	jeza %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**57.4**	11.9	0	5.6	11.9	10.2	2.9
jeza	27.3	**32.7**	0	8.3	14.7	11.2	5.8
gnus	13.8	44.8	**0**	0	27.6	10.3	3.4
strah	25.8	16.3	0	**16.6**	17.3	11.0	13.1
veselje	8.0	3.4	0	1.9	**82.1**	1.9	2.7
žalost	33.1	18.3	0	9.1	14.2	**19.2**	6.0
presenečenje	17.5	3.9	0	3.9	7.9	3.2	**63.6**

Tabela 4.11: Matrika pravilnih in napačnih razvrstitev (SVM z linearnim jedrom, Kaggle, podatki obdelani z LBP, klasifikacijska točnost 50,1%).

RBF-jedro Klasifikacijska točnost algoritma, ocenjenega na neodvisni testni množici, je 87,9 %. Ta klasifikator se je odrezal najbolje od vseh testiranih klasifikatorjev. Tudi v matriki pravilnih in napačnih razvrstitev (tabela 4.12) lahko vidimo, da je klasifikator tudi zelo izrazite obrazne izraze napovedal najbolje (jeza, gnus, veselje in presenečenje), manjše težave pa je imel s prezirom in strahom. Velikost samega klasifikatorja je 38,8 MB, hitrost napovedovanja pa je 409 primerov na sekundo. Klasifikator, naučen s podatkovno bazo Kaggle, je dosegel klasifikacijsko točnost 57,3 %. V matriki pravilnih in napačnih klasifikacij v tabeli 4.13 lahko vidimo, da je klasifikator z veliko klasifikacijsko točnostjo napovedal izraz presenečenje (68,6 %) in veselje (81,7 %), kar sta zelo izrazita obrazna izraza, manj pogosto pa je napovedal strah (36,7 %). Velikost klasifikatorja je 361,7 MB, hitrost napovedovanja pa je 36 primerov na sekundo.

4.4 Primerjava lastnosti naučenih klasifikatorjev

Poleg točnosti klasifikatorjev nas zanima tudi velikosti naučenega klasifikatorja in hitrost napovedovanja primerov.

V tabelah 4.14 in 4.15 so zapisane ključne lastnosti testiranih klasifikatorjev. Če primerjamo hitrost napovedovanja, lahko vidimo, da je neposredno odvisna od

	nevtralni %	jeza %	prezir %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**100.0**	0	0	0	0	0	0	0
jeza	0	**86.7**	0	0	0	0	13.3	0
prezir	66.7	0	**33.3**	0	0	0	0	0
gnus	0	0	0	**100.0**	0	0	0	0
strah	50.0	0	0	0	**50.0**	0	0	0
veselje	0	0	0	4.8	0	**95.2**	0	0
žalost	11.1	11.1	0	0	0	0	**66.7**	11.1
presenečenje	14.8	0	0	3.7	0	0	0	**81.5**

Tabela 4.12: Matrika pravilnih in napačnih razvrstitev (SVM z RBF-jedrom, Cohn-Kanade, podatki obdelani z LBP, klasifikacijska točnost 87,9 %).

	nevtralni %	jeza %	gnus %	strah %	veselje %	žalost %	presenečenje %
nevtralni	**49.9**	10.5	0	6.3	9.7	20.9	2.7
jeza	12.9	**46.4**	0	7.2	9.4	22.7	1.4
gnus	6.9	37.9	**37.9**	0	3.4	6.9	6.9
strah	14.1	11.0	0	**36.7**	8.8	21.6	7.8
veselje	5.3	3.2	0	1.7	**81.7**	6.1	1.9
žalost	21.1	15.5	0	8.8	10.7	**41.3**	2.5
presenečenje	5.7	4.3	0	4.6	4.6	12.1	**68.6**

Tabela 4.13: Matrika pravilnih in napačnih razvrstitev (SVM z RBF-jedrom, Kaggle, podatki obdelani z LBP, klasifikacijska točnost 57,3 %).

velikosti klasifikatorja. Hitrosti so bile izmerjene na prenosnem računalniku, na telefonu iPhone 5s pa so te hitrosti do 3,4-krat nižje. Najpočasneje napovedujejo zelo veliki klasifikatorji (naučeni z neobdelanimi podatki), klasifikacijska točnost pa je odvisna predvsem od izbrane metode strojnega učenja (SVM se je izkazal bolje kot k-NN) ter predhodne obdelave podatkov. Poleg tega je podatkovna baza Kaggle drugače zasnovana od podatkovne baze Cohn-Kanade. V podatkovni bazi Kaggle so predstavljeni obrazi oseb različnih starosti (od dojenčkov do starejših) in različnih ras. Obrazi tudi niso vsi enako obrnjeni in enako osvetljeni. Zaradi tega dajejo klasifikatorji, naučeni s podatkovno bazo Cohn-Kanade, veliko boljše rezultate od klasifikatorjev, naučenih s podatkovno bazo Kaggle. Za uporabo

Klasifikator	Velikost (MB)	Hitrost napovedovanja (primeri/s)	Klasifikacijska točnost (%)
neobdelani podatki			
k-NN	165,4	41	33,3
SVM (linearni)	12,3	2.252	70,4
SVM (RBF)	174,5	39	25.0
podatki obdelani z LBP			
k-NN	49,8	660	37,9
SVM (linearni)	1,6	16.154	87,1
SVM (RBF)	38,8	409	87,9

Tabela 4.14: Lastnosti naučenih klasifikatorjev (podatkovna baza Cohn-Kanade)

Klasifikator	Velikost (MB)	Hitrost napovedovanja (primeri/s)	Klasifikacijska točnost (%)
neobdelani podatki			
k-NN	1.804	4	41,9
SVM (linearni)	9,2	3.165	27,4
SVM (RBF)	205,5	23	25,0
podatki obdelani z LBP			
k-NN	453,1	28	44, 9
SVM (linearni)	1,2	21.411	50,1
SVM (RBF)	361,7	36	57,3

Tabela 4.15: Lastnosti naučenih klasifikatorjev (podatkovna baza Kaggle)

na prenosnih telefonih je najbolj primeren klasifikator z linearnim jedrom, ki se je predhodno naučil iz podatkovne baze Cohn-Kanade, s predhodno obdelanimi slikami s pomočjo LBP-algoritma, čeprav je klasifikator z RBF-jedrom (baza Cohn-Kanade) malenkost boljši (če primerjamo klasifikacijsko točnost vseh naučenih klasifikatorjev).

4.5 Primerjava z ostalimi algoritmi

Shan, Gong in McOwan [24] so izvedli podobne teste na podatkovni bazi Cohn-Kanade z LBP-algoritmom. Izbrali so 1280 različnih slik, jih obrezali na velikost 110 × 150 pik in poravnali glede na lokacijo oči (naše slike so bile velike 150 × 150 in neporavnane). Uporabili so podoben uniformi algoritem LBP s 59 koši, le da so sliko razdelili na 6 × 7 območij (naše slike smo razdelili na 7 × 7 območij). Za ocenjevanje rezultatov so uporabili metodo prečnega preverjanja. Pri napovedovanju le sedmih obraznih izrazov (nevtralni, gnus, jeza, veselje, presenečenje, žalost, strah) z linearnim in RBF-klasifikatorjem so dosegli podobno klasifikacijsko točnost kot klasifikatorji, naučeni v tem delu: 88,1 % in 88,9 %.

4.6 Pomanjkljivosti in prostor za izboljšave

Dosežena 87,9 % klasifikacijska točnost je dovolj dobra za različne aplikacije, ki so namenjene kratkočasenju in zabavi. To klasifikacijsko točnost bi bilo treba izboljšati, če bi želeli algoritem uporabljati za profesionalne namene, na primer v trgovinah ali pisarnah, predvsem zato, ker je ta klasifikacijska točnost izračunana s pomočjo podatkov iz "idealne"baze podatkov (Cohn-Kanade). Verjetno bi se v manj ugodnih razmerah (slaba osvetlitev, neporavnan obraz, dojenčkov obraz) najboljši klasifikator, naučen s podatkovno bazo Kaggle, odrezal bolje.

Pogosti razlogi za slabe rezultate so:

- premajhna slika obraza,
- slabo osvetljena slika,
- premalo izrazit obrazni izraz,
- različne človeške rase.

Naš algoritem bi lahko izboljšali z uporabo umetnih nevronskih mrež (družba Noldus [13] navaja, da so z uporabo umetnih nevronskih mrež dosegli natančnost več kot 90 %) in z uporabo bolj kakovostne, večje učne množice (podatkovna baza Kaggle je sicer velika, a je sestavljena iz zelo nekakovostnih, majhnih slik, ki zelo otežujejo klasifikacijo). Pred samim učenjem klasifikatorjev bi bilo treba

obrazne slike tudi poravnati in jim spremeniti velikost, da bi bila razdalja med očesoma vedno enaka. Poleg tega bi lahko sliko prej obdelali z več različnimi algoritmi za obdelavo podatkov. Najprej bi na primer uporabili aktivni model videza (angl. active appearance model) in potem uporabili LBP za določanje globalne teksture [40]. Uporabili bi lahko tudi več klasifikatorjev hkrati in na ta način dosegli boljšo točnost [42]. Ker pa smo se omejili le na prenosne telefone, smo omejeni z zmogljivostjo naprave, ki je še vedno tudi do 10-krat manjša od povprečnih računalnikov.

Poglavje 5

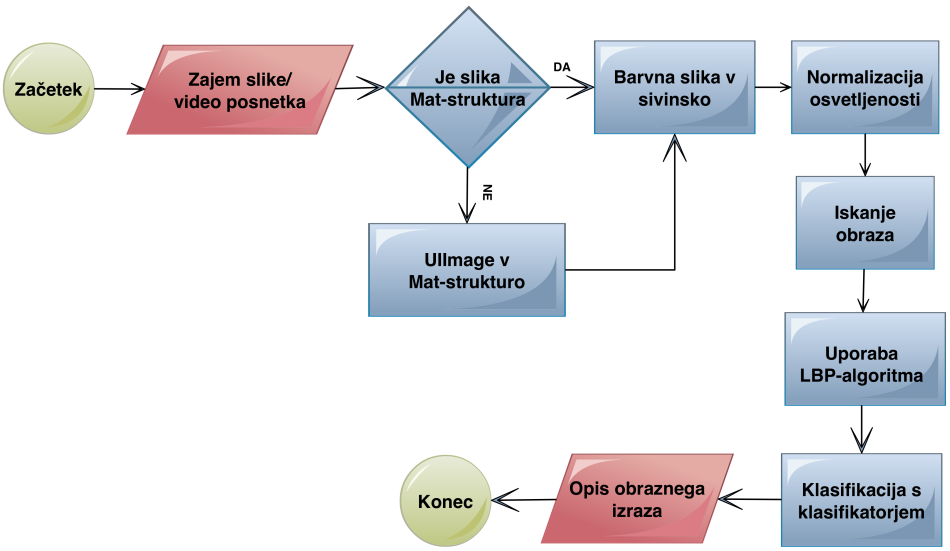
Priprava knjižnice za iOS napravo

Končni rezultat tega dela je enostavna knjižnica za uporabo na prenosnih telefonih z operacijskim sistemom iOS. Knjižnica omogoča zajemanje video posnetka iz kamere in avtomatsko označevanje prepoznanih obraznih izrazov. Razvijalec lahko knjižnico priloži k svojemu projektu in z uporabo metod v knjižnici pridobi podatke o obraznem izrazu na sliki ali video posnetku.

5.1 Struktura knjižnice

V različici 0.1 knjižnica omogoča razvijalcu prepoznavo obraznega izraza na eni sami sliki ali zaporedju slik. Knjižnica je prilagojena za delo na prenosnih napravah in je pripravljena za delo z video posnetki. Glavni koraki algoritma za napovedovanje izrazov v knjižnici so prikazani na sliki 5.1. Vsak korak rešuje manjši podproblem, vsi koraki pa so potrebni za rešitev celotnega problema:

- **Zajem slike ali video posnetka:** Razvijalec lahko uporabi kamero na prenosni napravi ali uporabi slike/video posnetke, shranjene na sami napravi.
- **Pretvorba slike/video posnetka v Mat-strukturo:** Zaradi lažje prenosljivosti med platformami mora razvijalec sam pretvoriti objekt UIImage/NSImage v Mat-strukturo (metoda za pretvorbo je na voljo v knjižnici).



Slika 5.1: Prikaz strukture knjižnice

- **Pretvorba slike v sivinsko sliko:** Vsi algoritmi so naučeni s sivinskimi slikami, zato je treba vse vhodne slike pretvoriti v sivinske slike.

- **Normalizacija osvetljenosti na sliki:** Z algoritmom za normalizacijo osvetljenosti popravimo kontraste na sliki.

- **Iskanje obraza:** Z algoritmom Viola-Jones poiščemo obraze na sliki. Vedno uporabimo prvi zaznani obraz (algoritem Viola-Jones poskusi poiskati vse obraze na sliki).

- **Uporaba LBP-algoritma:** Podatke na sliki preoblikujemo z LBP-algoritmom.

- **Napovedovanje obraznega izraza:** Preoblikovane podatke posredujemo klasifikatorju, ki nato napove obrazni izraz.

- **Opis obraznega izraza:** Izhod iz algoritma je opis obraznega izraza (niz besed).

5.2 Javni vmesnik za delo s knjižnico

Razvijalec lahko knjižnico uporablja preko javnega vmesnika, tako da v razred, kjer želi uporabljati knjižnico, vključi (angl. import) datoteko *iEmotions.h*. V razredu *iEmotions* sta na voljo dve javni metodi:

- **syncProcessImage:** Sinhrona metoda za procesiranje slik. Deluje na glavni niti in prekine izvajanje aplikacije, dokler ne vrne rezultata (neprimerna za video posnetke).
- **asyncProcessImage:** Asinhrona metoda za procesiranje slik. Deluje na stranski niti in ne prekine izvanja aplikacije. Rezultat vrne razvijalcu v blokovni funkciji, ko končna s procesiranjem (primerna tudi za video posnetke).

Obe metodi vračata rezultat v blokovni funkciji (angl. block). Blokovna funkcija vrne opis obraznega izraza (niz znakov) in sliko zaznanega obraznega izraza (struktura OpenCV Mat).

5.2.1 Opis glavnih delov knjižnice

Glavni razred aplikacije

Glavni razred aplikacije, imenovan *MKFaceExpressionManager*, skrbi za celotno delovanje knjižnice. Hkrati lahko obstaja le en objekt tega razreda (angl. Singleton). Ob klicu katerekoli javne metode se ustvari en primerek tega razreda (če še ne obstaja). Ob stvaritvi objekta se naloži SVM-klasifikator za napovedovanje obraznih izrazov in klasifikator Viola-Jones za iskanje obrazov na slikah. Nalaganje klasifikatorjev z diska je počasna operacija [44], zato je pomebno, da se klasifikatorja naložita v pomnilnik telefona ob prvi uporabi knjižnice. 1,2 MB velika datoteka se na telefonu iPhone 5S nalaga 0,0423 sekunde, ko pa je enkrat naložena v pomnilnik, je čas dostopa zanemarljiv.

Asinhrona metoda za napovedovanje obraznih izrazov

Ta metoda sprejme strukturo OpenCV Mat, ki je predstavljena kot matrika v dveh dimenzijah, in blokovno funkcijo, ki vrača niz s prepoznanim obraznim izrazom

ter območje zaznanega obraza v strukturi OpenCV Mat. Ob klicu te metode se ne blokira nit izvajanja, na kateri je bila metoda klicana. Metoda vrne rezultat procesiranja takoj, ko zaključi s prepoznavo obraznega izraza. Rezultat je shranjen v parametrih blokovne funkcije in je dostopen na glavni niti. Metodo lahko kličemo tako hitro, kot zajemamo slike za video posnetek (ponavadi s 24 slikami na sekundo), metoda pa sprejme naslednjo sličico, ko konča s procesiranjem prejšnje. Tak način procesiranja slik nam omogoča, da kljub manj zmogljivim CPE v prenosnih telefonih še vedno napovedujemo obrazne izraze na video posnetku brez zatikanja in izgubljanja slik med samim predvajanjem posnetka (v času pisanja tega dela je na tržišču na voljo telefon iPhone 5S z dvojedrnim 1,4Ghz A7 procesorjem, ki za obdelavo ene slike potrebuje 0,33 sekunde).

5.3 Priprava statične knjižnice

Na operacijskem sistemu iOS imamo na voljo dve različni obliki knjižnic: statična knjižnica [43] in programersko ogrodje (angl. framework) [45]. Prednost programskega ogrodja pred statično knjižnico je, da nam ni treba posebej vključevati datotek .h, ki vsebujejo deklaracije javnih metod. Slabost pa je, da je ogrodje težje pripraviti. V tem delu smo se zaradi večje razširjenosti uporabe tega načina in enostavnejše priprave odločili za uporabo statične knjižnice. Poleg tega je v trenutni dokumentaciji družbe Apple to tudi edini uradno podprti način.

Ker smo želeli, da je knjižnica majhna, smo razdelili programsko kodo na dva dela: na del za prepoznavanje obraznih izrazov in na del za testiranje učinkovitosti algoritma. V javno knjižnico smo vključili le del za prepoznavanje obraznih izrazov.

V času pisanja tega dela so na tržišču naprave z ARM-arhitekturo armv7 in armv7s (32-bitni procesorji) ter arm64 (64-bitni procesorji). Zato je treba pripraviti knjižnico za vsaj te tri arhitekture. Poleg tega lahko razvijalec programsko opremo testira tudi znotraj vgrajenega simulatorja, ki deluje na računalnikih Mac z 32-bitnimi in 64-bitnimi procesorji. To pomeni, da je treba knjižnico pripraviti tudi za ti dve arhitekturi.

5.3.1 Postopek za novo tarčo s statično knjižnico

Za pripravo statične knjižnice z eno arhitekturo je bilo treba slediti spodnjim korakom:

1. **Izdelava nove tarče:** Znotraj aplikacije Xcode smo dodali novo tarčo (angl. Target) za izdelek (angl. Product), ki je namenjena izdelavi statične knjižnice.

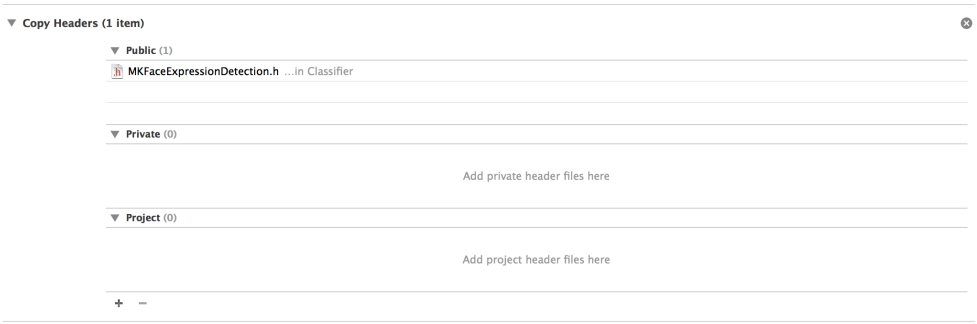
2. **Določitev izvorne kode za novo tarčo:** Izbrali smo vse datoteke z izvorno kodo, ki smo jih želeli vključiti v naš projekt. V knjižnico ne smemo vključiti zunanjih knjižnic in programerskih ogrodij, kot je npr. OpenCV, ker so lahko zaščitene z licencami. Poleg tega moramo razvijalcu pustiti nadzor, katere knjižnice in programerska ogrodja bo vključil v svojo aplikacijo.

3. **Določitev javnih metod:** Ko smo dodali vse izvorne datoteke v knjižnico, je bilo treba določiti tudi, katere .h-datoteke z javnimi metodami bodo vidne razvijalcu. To smo naredili, tako da smo dodali korak, ki skopira .h-datoteke poleg naše statične knjižnice (slika 5.2). Če želi razvijalec uporabljati našo knjižnico, mora svojemu projektu priložiti te .h-datoteke ali v projektu nastaviti pot do teh datotek.

S tem je bil osnovni postopek priprave knjižnice končan. Vendar smo s tem pripravili le tarčo za izdelavo knjižnice za eno arhitekturo in brez drugih datotek z viri (slike, naučeni klasifikatorji ipd.).

5.3.2 Postopek za pripravo univerzalne statične knjižnice

Za izdelavo univerzalne knjižnice (kjer je vključenih vseh 5 arhitektur), je treba uporabiti orodje Lipo, ki ga poženemo znotraj terminalskega okna. Za ta namen smo napisali skripto v jeziku Bash, ki pripravi več knjižnic za vsako arhitekturo posebej, nato pa program Lipo te združi v eno univerzalno knjižnico (koda A.1 v poglavju priloge).



Slika 5.2: Določitev javno dostopnih .h datotek v projektu

Koraki v skripti

1. Zgradili smo različice statične knjižnice za 3 različne arhitekture (arm, i386, x86_64). Pri tem smo uporabili tarčo, ki smo jo pripravili po zgoraj opisanem postopku.

2. S programom Lipo smo združili vse 3 knjižnice v eno univerzalno.

3. Skopirali smo .h-datoteke v mapo include, ki se nahaja zraven naše nove univerzalne knjižnice.

5.3.3 Priprava paketa z datotekami

Ko smo pripravili statično knjižnico, je bilo treba zapakirati tudi ostale datoteke. V našem primeru je treba razvijalcu poleg knjižnice ponuditi tudi datoteki z naučenim klasifikatorjem Viola-Jones za iskanje obrazov na sliki in SVM-klasifikatorjem za prepoznavanje obraznih izrazov. To lahko storimo z uporabo paketa z datotekami (angl. Bundle).

Postopek priprave

Znotraj aplikacije Xcode smo dodali novo tarčo (angl. Target) za izdelek (angl. Product), ki je namenjena izdelavi paketa z datotekami. Nato smo izbrali vse datoteke, kjer so shranjeni klasifikatorji, in jih dodali koraku, ki skopira datoteke v paket (angl. copy bundle resources).

5.4 Različne optimizacije za zmogljivost prenosnih telefonov

Ker je opravilo prepoznave obraznih izrazov procesorsko zelo zahtevno, je treba nekatere dele programske kode poenostaviti in uporabiti kodo, ki izkorišča lastnosti centralnoprocesnih enot v mobilnih telefonih.

5.4.1 Uporaba kode, ki izkorišča lastnosti CPE

Družba Apple je pripravila programersko ogrodje *Accelerate*, ki pohitri vektorske operacije, poleg tega pa je ta koda, kjer je to le mogoče, tudi strojno pospešena. V naši knjižnici smo to programersko ogrodje uporabili za pretvorbo barvnih slik v sivinske in za normalizacijo osvetljenosti [2]. Na izseku kode 5.1 je prikazan algoritem, ki deluje s programerskim ogrodjem Accelerate.

Izvorna koda 5.1: Koda Accelerate za pretvarjanje barvnih slik v sivinske

```
long cvtColor_Accelerate(const cv::Mat& src, const cv::Mat& dst)
{
    vImagePixelCount rows = static_cast<vImagePixelCount>(src.rows);
    vImagePixelCount cols = static_cast<vImagePixelCount>(src.cols);

    cv::Mat accBuffer1;
    accBuffer1.create(src.size(), src.type());
    cv::Mat accBuffer2;
    accBuffer2.create(src.size(), CV_8UC1);

    vImage_Buffer _src = { src.data, rows, cols, src.step};
    vImage_Buffer _dst = { dst.data, rows, cols, dst.step};
    vImage_Buffer buff1 = {accBuffer1.data, rows, cols, accBuffer1.step};
    vImage_Buffer buff2 = {accBuffer2.data, rows, cols, accBuffer2.step};

    const int16_t matrix[4 * 4] = {  77, 0, 0, 0,
        150, 0, 0, 0,
        29, 0, 0, 0,
        0, 0, 0, 0 };
    int32_t divisor = 256;

    vImage_Error err;
    err = vImageMatrixMultiply_ARGB8888(&_src, &buff1,
                                        matrix, divisor,
                                        NULL, NULL, 0 );
```

```
    err = vImageConvert_ARGB8888toPlanar8(&buff1, &_dst,
                                          &buff2, &buff2, &buff2, 0);
    return err;
}
```

Tudi sama knjižnica OpenCV že vključuje nekatere metode, ki so strojno pospešene in temeljijo na programerskem ogrodju Accelerate (nekateri DSP-filtri) [46].

5.4.2 Poenostavitve

Tudi če bi kodo pohitrili povsod, kjer nam nivo paralelizma ali matematika to dopuščata, še vedno na nekaterih starejših napravah prepoznavanje obraznih izrazov ob zajemu video posnetkov pri 24 slikah na sekundo ne bi delovalo tekoče. Zato je treba uporabiti nekatere poenostavitve.

Vemo, da človeško oko ne loči posameznega intervala med slikami v video posnetkih. Zaradi tega tudi ni smiselno, da bi osveževali podatke o trenutnem obraznem izrazu vsako 1/24 sekunde. To bi lahko naredili na primer vsako sekundo in uporabnik bi še vedno dobil dovolj točne, čeprav ne tako tekoče podatke.

V naši knjižnici je to rešeno tako, da se takoj, ko je na voljo nova slika iz kamere, slika pošlje v obdelavo. Slika se bo obdelala le, če se trenutno ne obdeluje nobena druga slika. V nasprotnem primeru se poslane slike ne upošteva in se počaka na novo. Na izseku kode 5.2 je prikazan algoritem za asinhrono obdelavo slik, ki implementira opisano lastnost.

Izvorna koda 5.2: Koda za asinhrono procesiranje slik

```
// Public methods
- (void)asyncProcessImage:(Mat)image
                  completionBlock:(void (^)(NSString *expressionLabel, Mat
                      detectedFaceRegion))completionBlock
{
        if (imageIsBeingProcessed) return;
        imageIsBeingProcessed = YES;

        Mat copiedImage;
        image.copyTo(copiedImage);
        dispatch_async(dispatch_get_global_queue(
            DISPATCH_QUEUE_PRIORITY_HIGH, 0ul), ^{
                cv::Mat capturedImage;

                cv::transpose(copiedImage, capturedImage);
                cv::flip(capturedImage, capturedImage, 1);

                Mat detectedFaceMat;
                NSString *expressionText = [[MKFaceExpressionDetection
                    sharedInstance] detectExpressionOnImage:capturedImage
                    detectedFace:detectedFaceMat];

                dispatch_async(dispatch_get_main_queue(), ^{
                        completionBlock(expressionText, Mat());
                        imageIsBeingProcessed = NO;
                });
        });
}
```

Poglavje 6

Sklepne ugotovitve

V tem delu je predstavljena knjižnica za samodejno prepoznavanje obraznih izrazov, ki deluje na prenosnih telefonih z operacijskim sistemom iOS. Omogoča prepoznavo izrazov na fotografijah in video posnetkih skoraj v realnem času (na telefonu iPhone 5S traja prepoznavanje slike 0,422 sekunde). Končni izdelek tega dela je knjižnica za uporabo v različnih aplikacijah, ki lahko obdelujejo fotografije in video posnetke. Algoritem za prepoznavo obraznih izrazov je zasnovan tako, da je prenosljiv med platformami. Knjižnica je bila razvita z namenom:

- predstaviti enostaven način za prepoznavo obraznih izrazov,
- poenotiti knjižnico za različne mobilne platforme.

Algoritem je sestavljen iz več korakov, ki nam omogočajo razmeroma točno prepoznavo obraznega izraza na sliki:

1. Obrazni izraz na sliki poiščemo in označimo z uporabo algoritma Viola-Jones.
2. Z LBP-algoritmom podatke preoblikujemo v primernejšo obliko.
3. Obrazni izraz prepoznamo s pomočjo predhodno naučenega klasifikatorja z metodo podpornih vektorjev (SVM).

Razvita programska oprema uporablja različne metode računalniškega vida, za določene optimizacije pa je uporabljena tehnologija družbe Apple, ki je vgrajena neposredno v jedro operacijskega sistema iOS. Učinkovitost samega algoritma je

dovolj dobra za amatersko uporabo (v raznih igrah in zabavnih aplikacijah), za profesionalno rabo pa bi bilo treba algoritem še dodatno izboljšati. Zaradi razmeroma majhnih zmogljivosti prenosnih telefonov pa smo tudi bolj omejeni, kot če bi razvijali algoritem za uporabo na namiznih računalnikih.

Dodatek A

Koda za izdelavo univerzalne knjižnice

Dodatek A.1 prikazuje le izsek kode, ki iz tarče knjižnice ustvari univerzalno knjižnico. Tako pripravljena knjižnica lahko deluje na vseh petih podprtih arhitekturah (armv7, armv7s, arm64, i386 ter x86_64).

Izvorna koda A.1: Koda Bash za izdelavo univerzalne knjižnice

```
# define output folder environment variable
UNIVERSAL_OUTPUTFOLDER=${BUILD_DIR}/${CONFIGURATION}-universal

LIBRARY_NAME ="iEmotions"

# Step 1. Build Device and Simulator versions
xcodebuild -target faceExprRecognition ONLY_ACTIVE_ARCH=NO -configuration ${
    CONFIGURATION} -sdk iphoneos  BUILD_DIR="${BUILD_DIR}" BUILD_ROOT="${
    BUILD_ROOT}" PRODUCT_NAME="iEmotions"
xcodebuild -target faceExprRecognition -sdk 'iphonesimulator7.0' -
    configuration ${CONFIGURATION}  clean build ARCHS='x86_64' VALID_ARCHS='
    x86_64' IPHONEOS_DEPLOYMENT_TARGET='7.0' BUILD_DIR="${BUILD_DIR}"
    BUILD_ROOT="${BUILD_ROOT}" PRODUCT_NAME="iEmotions_x86_64"
xcodebuild -target faceExprRecognition -sdk 'iphonesimulator7.0' -
    configuration ${CONFIGURATION}  clean build ARCHS='i386' VALID_ARCHS='
    i386' IPHONEOS_DEPLOYMENT_TARGET='7.0' BUILD_DIR="${BUILD_DIR}"
    BUILD_ROOT="${BUILD_ROOT}" PRODUCT_NAME="iEmotions_i386"


# make sure the output directory exists
mkdir -p "${UNIVERSAL_OUTPUTFOLDER}"
```

```
# Step 2. Create universal binary file using lipo
lipo -create -output "${UNIVERSAL_OUTPUTFOLDER}/libiEmotions.a" "${BUILD_DIR
    }/${CONFIGURATION}-iphoneos/libiEmotions.a" "${BUILD_DIR}/${
    CONFIGURATION}-iphonesimulator/libiEmotions_i386.a" "${BUILD_DIR}/${
    CONFIGURATION}-iphonesimulator/libiEmotions_x86_64.a"

# Last touch. copy the header files. Just for convenience
cp -R "${BUILD_DIR}/${CONFIGURATION}-iphoneos/usr/local/include" "${
    UNIVERSAL_OUTPUTFOLDER}/"
```

Slike

Tabele

Literatura

[1] A. Kaehler, G. Bradski, *Learning OpenCV*, Kalifornija, ZDA: O'Reilly Media, 2013

[2] K. Kornyakov, A. Shishkov, *Instant OpenCV for iOS*, Birmingham, Velika Britanija: Packt Publishing, 2013

[3] Y.L. Tian, T. Kanade, J. F. Cohn, *Handbook of Face Recognition*, New York, ZDA: Springer Science+Business Media, Inc., 2005

[4] (2014) The Facial Expression Coding System (FACES) Dostopno na: `http://ist-socrates.berkeley.edu/~akring/FACES%20manual.pdf`

[5] (2014) iOS Tehnology Overview. Dostopno na: `https://developer.apple.com/library/ios/documentation/`

[6] (2014) About OpenCV. Dostopno na: `http://opencv.org`

[7] (2014) Detecting Faces in an Image. Dostopno na: `https://developer.apple.com/library/ios/documentation/GraphicsImaging/Conceptual/CoreImaging/ci_detect_faces/ci_detect_faces.html`

[8] (2014) FaceDetector – Android. Dostopno na: `http://developer.androidcom/reference/android/media/FaceDetector.Face.html`

[9] (2014) Software and labs for behavioral research and video tracking — Noldus Information Technology. Dostopno na `http://www.noldus.com`

[10] (2014) Percipo: Video Analysis for Retail Analytics : Foot Traffic, Age Estimation and Gender Identification. Dostopno na: `http://www.percipo.com/`

[11] (2014) Advanced Source Code – Facial Expression Recognition System. Dostopno na: http://www.advancedsourcecode.com/facialexpression.asp

[12] (2014) Source Code Online – Facial Expression Recognition Matlab. Dostopno na: http://www.sourcecodeonline.com/list?q=facial_expression_recognition_matlab

[13] (2014) FaceReader methodology. Dostopno na: ttp://www.noldus.com/webfm_send/618

[14] (2014) What is Face Description. Dostopno na: http://www.percipo.com/news.html

[15] (2014) Photo Age – How Old Do You Really Look in That picture? on the App Store in iTunes. Dostopno na: https://itunes.apple.com/ie/app/photoage-how-old-do-you-really/id391310821?mt=8

[16] J.F. Cohn, Z. Ambadar, P. Ekman, *The Handbook of Emotion Elicitation and Assessment*, Oxford University Press: New York, NY, 2007

[17] (2014) Face Detection Homepage – Techniques. Dostopno na: http://www.facedetection.com/facedetection/techniques.htm

[18] P. Viola, M. J. Jones, Robust Real-Time Face Detection, *International Journal of Computer Vision*, 57(2), str. 137–154, 2004

[19] B. Menser, F. Müller, Face Detection In Color Images Using Principal Components Analysis, *Seventh Int'l Conf. on Image Processing and Its Applications*, str. 620–624, 1999

[20] K. Kollreider, H. Fronthaler, M. I. Faraj, J. Bigun, Real-time Face Detection and Motion Analysis with Application in "Liveness" Assessment, *IEEE Transactions on information forensics and security, 10(10)*, 2007

[21] (2014) Face Detection using Haar Cascades. Dostopno na: http://docs.opencv.org/trunk/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html

[22] S. Z. Li, A. Jain, *Encyclopedia of Biometric*, New York, ZDA: Springer, 2009

[23] L. S. Lopez, Local Binary Patterns applied to Face Detection and Recognition, zaključni projekt, Escola Tecnica Superior d'Enginyeria de Telecomunicacio de Barcelona, Universitat Politecnica De Catalunya, 2010

[24] C. Shan, S. Gong, P. W. McOwan, Facial expression recognition based on Local Binary Patterns: A comprehensive study', *Image and Vision Computing*, 27, str. 803–816, 2009

[25] T. Ojala, M. Pietikäinen, T. Mäenpää, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, (7), str. 971–987, 2002

[26] T. Ojala, M. Pietikainen, D. Harwood, A comparative study of texture measures with classification based on featured distribution, *Pattern Recognition*, 29, str. 51–59, 1996

[27] (2014) A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm. Dostopno na: `http://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/`

[28] (2014) Github - knearest.cpp. Dostopno na: `https://github.com/Itseez/opencv/blob/master/modules/ml/src/knearest.cpp#L181`

[29] (2014) Neghbourhood Component Analysis. Dostopno na: `http://www.cs.toronto.edu/~fritz/absps/nca.pdf`

[30] J. Bergstra, Y. Bengio, Random Search for Hyper-Parameter Optimization, *Journal of Machine Learning Research 13*, str. 281–305, 2012. Dostopno na: `http://jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf`

[31] StatSoft – k-Nearest Neighbors. Dostopno na: `http://www.statsoft.com/textbook/k-nearest-neighbors`

[32] D. Huson, Algorithms in Bioinformatics II, *SoSe'07, ZBIT*, str. 263–279, 2007. Dostopno na: `http://ab.inf.uni-tuebingen.de/teaching/ss07/albi2/script/svm.pdf`

[33] (2014) Connectionist and Statistical Language Processing. Dostopno na: http://www.coli.uni-saarland.de/~crocker/Teaching/Connectionist/lecture11_4up.pdf

[34] (2014) I. H. Witten, F. Eibe, Credibility: Evaluating What's Been Learned, *Data Mining*, str. 147-187, San Francisco, ZDA: Morgan Kaufmann Publishers Inc., 2005. Dostopno na http://aml.media.mit.edu/EvaluationWitten&Frank.pdf

[35] I. Kononenko, *Strojno učenje*, Ljubljana, Slovenija: Založba FE in FRI, 2005

[36] (2014) Support Vector Machines (SVM) Introductory Overview. Dostopno na: http://www.statsoft.com/textbook/support-vector-machines

[37] O. Carugo, F. Eisenhaber, *Data Mining Techniques for the Life Sciences*, New York, ZDA: Springer, 2010

[38] (2014) The Affect Analysis Group at Pittsburgh. Dostopno na: http://www.pitt.edu/~emotion/ck-spread.htm

[39] (2014) Challenges in Representation Learning: Facial Expression Recognition Challenge. Dostopno na: http://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

[40] M. W. Huang, Z. W. Wang, Z. L. Ying, A novel method of facial expression recognition based on GPLVM Plus SVM, *2010 IEEE 10th International Conference on Signal Processing*, str. 916-919, 2010

[41] P. Gaspar, J. Carbonell, J. L. Oliveira, On the parameter optimization of Support Vector Machines for binary classification, *Journal of Integrative Bioinformatics*, 9(3), str. 201–211, 2012. Dostopno na: http://journal.imbio.de/articles/pdf/jib-201.pdf

[42] X. Huang, G. Zhao, M. Pietikäinen, W. Zheng, Dynamic Facial Expression Recognition Using Boosted Component-Based Spatiotemporal Features and Multi-Classifier Fusion. *Proceedings of Advanced Concepts for Intelligent Vision Systems*, Avstralija, str. 312–322, 2010

[43] (2014) Using Static Libraries in iOS: Creating the Library. Dostopno na: `https://developer.apple.com/library/ios/technotes/iOSStaticLibraries/Articles/creating.html`

[44] D. Kodek, Glavni pomnilnik in predpomnilniki, *Arhitektura in organizacija računalniških sistemov*, str. 267–318, Senčur, Slovenija: Bi-Tim, 2008

[45] (2014) Universal Framework for iOS. Dostopno na: `http://blog.db-in.com/universal-framework-for-ios/`

[46] (2014) OpenCV iOS – Video Processing – OpenCV 3.0.0-dev documentation. Dostopno na: `http://docs.opencv.org/trunk/doc/tutorials/ios/video_processing/video_processing.html`